UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO

DESARROLLO DE UNA AYUDA TÉCNICA PARA ALUMNOS DEL COLEGIO SAN RAFAEL (15): TRANSICIÓN A LA VIDA ADULTA – JUEGO DE CARRERAS

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

AUTOR: SERGIO CASTILLO MOHEDANO

TUTOR: RICARDO VERGAZ BENITO

LEGANÉS, 26 DE SEPTIEMBRE DE 2016

AGRADECIMIENTOS

Este proyecto marca la culminación de una de las etapas más desafiantes que he podido tener en toda mi vida. Espero que me permita abrir nuevas fronteras hacia etapas que sean igual, o más si cabe, de nutrientes tanto a nivel personal como profesional. Lo que me llevo de estos cinco años, desde vivencias personales hasta conocimientos, pasando por compañeros y amigos, lo guardaré durante toda la vida. No puedo más que dar las gracias a todas las personas que se han cruzado conmigo en esta parte del camino, tanto para bien como para mal.

Quiero en primer lugar dar las gracias al Grupo de Displays y Aplicaciones Fotónicas de la UC3M, en especial a mi tutor Ricardo, por hacer posible que un Proyecto de Fin de Grado pueda tener una finalidad tan noble. Gracias también a mi compañero de proyecto, a Sergio Aguilera. Gracias por tu paciencia ante mis divagaciones sobre cómo complicar aún más un proyecto que ya de por sí abarcaba mucho, y gracias por haber cedido un poquito.

En la consecución del proyecto han participado muchas personas sin cuya ayuda jamás hubiera sido posible terminarlo. Para empezar, me gustaría dar las gracias a mis compañeros de *Indra*, quienes desde un primer momento me ayudaron en cualquier duda relativa al proyecto. Gracias por haberme prestado desde conectores y otros componentes, hasta una fuente de alimentación y osciloscopio para poder avanzar desde casa.

Tampoco hubiera sido posible este proyecto sin la ayuda de la empresa de fabricación de circuitos electrónicos *2CISA*, quienes desde un primer momento se mostraron dispuestos a fabricar las dos placas de circuito impreso de forma completamente gratuita, siendo la calidad de las placas de nivel profesional.

Así mismo, quisiera dar las gracias a *SAMTEC*, quienes nos enviaron, como muestra y sin coste alguno, los terminales que conectan las placas de desarrollo del microcontrolador con las placas de circuito impreso.

Por último, quisiera dar las gracias a mi entorno personal. Empezando por esos amigos, que ya sabes que van a ser para toda una vida, que han estado apoyándome y dándome ese empuje que en ocasiones necesitaba.

A mi hermano Cristian, que mientras escribo estas líneas estás durmiendo a escasos metros y aguantándome una noche más, desde hace meses e incluso años de estudios. Gracias por tu infinita paciencia. A mi hermano Fran, por todo tu apoyo y cariño, por preguntarme cómo iba y si necesitaba cualquier tipo de ayuda, gracias por estar siempre ahí. A mis padres, sin cuya ayuda este proyecto tampoco hubiera sido posible. De ellos fue la idea de utilizar las correas que llevan los carriles. Gracias por dejarme ocupar el salón de casa y permitir que la habitación parezca un pequeño y caótico laboratorio.

A mi padre y a mi madre,
porque este paso mío
fueron cien suyos ayer.

Gracias.

RESUMEN

El presente trabajo que a continuación se expone forma parte de un marco común de colaboración que desde hace años mantienen el Colegio de Educación Especial del Hospital San Rafael y el Grupo de Displays y Aplicaciones Fotónicas de la Universidad Carlos III de Madrid (GDAF-UC3M).

El objetivo del proyecto es la construcción de un juego de carreras adaptado para los alumnos con discapacidad del colegio San Rafael en su etapa en la Transición a la Vida Adulta.

Para una visión completa del trabajo **"Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida adulta – Juego de Carreras"** es indispensable la lectura conjunta de este proyecto con el de Sergio Aguilera Sevilla. La carga de trabajo se divide entre los dos documentos, mientras que el presente documento describe los sistemas del Bloque Carriles y el diseño de las placas de circuito impreso, el proyecto de Sergio Aguilera Sevilla se centra en la descripción de los sistemas del Bloque Rampa y la fabricación mecánica de los chasis de madera.

La comunicación entre ambos bloques se implementa mediante radiofrecuencia. La comunicación es unidireccional, toda la información que se transmite es desde el Bloque Rampa hacia el Bloque Carriles.

El Bloque Carriles está formado por cuatro cintas transportadoras movidas por cuatro motores paso a paso, comandados por un microcontrolador que a su vez recibe órdenes mediante radiofrecuencia desde el microcontrolador situado en el Bloque Rampa.

El Bloque Rampa manda estas órdenes al Bloque Carriles en función del estado del juego en el que se encuentre. Bien pueden ser órdenes relativas a mover un motor una determinada distancia, o relativas a decir qué número de jugadores hay en la partida, o para indicar la finalización del juego. Durante todo el juego se produce un barrido auditivo de lo que sucede.

El juego ha sido probado en el Colegio San Rafael y a día de hoy permanece allí funcionando correctamente.

ABSTRACT

This Project exposed here is part of the collaboration between the Special Education School of the Hospital San Rafael in Madrid for children with cognitive and motor disability and the Displays and Photonics Applications Group of the Carlos III University of Madrid (GDAF-UC3M).

The aim of the project is the implementation of a races game adapted for pupils with disabilities of the Special Education School and help them in their Transition to Adult Life.

In order to achieve a global visión of the project **"Development of a technical help for San Rafael Special Education School students (15): Transition to Adult Life – Races Game.".** It is mandatory to read it jointly with Sergio Aguilera Sevilla's project. The work load is divided between these two documents. The present document describes the development of the Rails Block and the design of the printed circuit boards of both Rails Block and Ramp Block. Sergio Aguilera Sevilla's project describes the development of the Ramp Block and the construction process of the chasis of both Rails Block and Ramp Block.

Communication between the two blocks is possible because of the implementation of a radiofrequency line. The data flow is unidirectional, and all the stream is from Ramp Block to Rails Block.

Rails Block is composed by four "conveyer belts" moved by four stepper motors, these motors are commanded by a microcontroller which receives the orders from other microcontroller placed in the Ramp Block.

Ramp Block sends instructions to Rails Block depending on in which phase the game is. These orders can be related to move one of the stepper motors a specific distance, or maybe related to indicate how many players it will be on that match. During the duration of the game, there is always an auditory scanning of what is going on each time.

The game has been proven and tested in the school and at this very moment it is placed there, operating perfectly.

ÍNDICE

Índice de Ilustraciones. 10
Índice de Acrónimos. 12
1. Capítulo 1. Introducción y Objetivos. 14
1.1. Introducción. 14
1.2. Motivación. 14
1.2.1. Transición a la Vida Adulta. 14
1.2.2. Diseño para todos. 15
1.2.3. Colegio San Rafael. 16
1.3. Estado del Arte. 18
1.4. Objetivos. 20
1.5. Especificaciones del Sistema. 20
1.5.1. Requisitos del Sistema. 20
1.6. Metodología. 22
1.7. Fases del Proyecto. 23
1.8. Medios Utilizados. 24
1.9. Descripción de la memoria. 25
2. Capítulo 2. Diseño del Sistema. 28
2.1. Funcionamiento detallado del juego. 28
2.2. Diagrama de bloques del sistema. 32
2.3. Bloque Carriles. 35
2.3.1. Motores paso a paso. 35
2.3.2. Microrruptores. 40
2.3.3. Sistema de audio. 47
2.3.4. Sistema de recepción por radiofrecuencia. 49
2.3.5. Microcontrolador. 54
2.3.6. Alimentación. 57
2.4. Diseño de las placas de circuito impreso. 58
2.4.1. Conceptos y elementos básicos comunes a ambas placas. 58
2.4.2. Proceso de diseño de las placas. 64
3. Capítulo 3. Implementación del sistema. 66
3.1. Pruebas del Bloque Carriles. 66
3.1.1. Motores paso a paso. 66
3.1.2. Microrruptores. 67
3.1.3. Sistema de Audio. 67

3.1.4. Receptor de Radiofrecuencia. ... 70
3.2. Firmware. ... 72
3.2.1. Firmware del Bloque Carriles. ... 73
3.3. Fabricación y ensamblaje. ... 79
4. Capítulo 4. Pruebas y resultados experimentales. ... 84
4.1. Pruebas. ... 84
4.2. Pruebas de campo. ... 88
5. Conclusiones, posibles mejoras y líneas futuras. ... 90
5.1. Conclusiones. ... 90
5.2. Mejoras. ... 90
5.3. Líneas futuras. ... 90
6. Presupuesto. ... 92
Bibliografía ... 96
Anexos. ... 100

Índice de Ilustraciones.

Ilustración 1. Colegio de Educación Especial San Rafael. Fachada [Fotografía]. (5) 16
Ilustración 2. Pictograma Arasaac en acceso a sala Logopedia [Fotografía]. (5) 17
Ilustración 3. Juego de Carreras [Imagen]. (7) 18
Ilustración 4. El 39% de los juguetes no son adecuados para ser usados por personas con discapacidad motora. [Gráfico]. (30) 18
Ilustración 5. A medida que aumenta el rango de edad al que va dirigido el producto, el porcentaje de juguetes adaptados disminuye. [Gráfico]. (30) 19
Ilustración 6. Solo el 5% de los productos analizados cumplen con las pautas del "Diseño para Todos". [Gráfico]. (30) 19
Ilustración 7. De Prado Escudero, M. (2013) Pulsador adaptado para colocar en silla de ruedas que puede ser ajustado en altura y posición. [Fotografía]. (31) 20
Ilustración 8. Laboratorio 1.2.C12. [Fotografía] 24
Ilustración 9. Detalle de fuentes de alimentación y osciloscopio, equipo portátil y placa de Bloque Carriles. [Fotografía]. 24
Ilustración 10. Mensaje en LCD: "Seleccione número de jugadores." [Fotografía]. 28
Ilustración 11. Bloque Rampa. La bola cae de la caseta de la parte superior. [Fotografía]. 29
Ilustración 12. Casillas del Bloque Rampa. Cada casilla tiene una puntuación. [Fotografía] 29
Ilustración 13. Un carril del Bloque Carriles. Su longitud es de 70 cm. [Fotografía] 30
Ilustración 14. Mensaje en LCD: "Pulse OK para continuar." [Fotografía]. 30
Ilustración 15. Mensaje en LCD: "Jugador 1 gana con 85 puntos." [Fotografía]. 31
Ilustración 16. Mensaje en LCD: "Pulse RESET para nueva partida." [Fotografía}. 31
Ilustración 17. Diagrama de bloques del Bloque Rampa. [Diagrama]. 32
Ilustración 18. Panel de control, [Fotografía]. 33
Ilustración 19. Botón Dome Rojo [Fotografía]. (9) 33
Ilustración 20. Diagrama de bloques del Bloque Carriles [Diagrama]. 34
Ilustración 21. Motor bipolar paso a paso 42BYGHW811. [Fotografía] 35
Ilustración 22. Motor bipolar. [imagen]. (10) 36
Ilustración 23. Driver A4988 [Fotografía] 36
Ilustración 24. Esquemático del driver StepStick de RepRap [Esquema]. 37
Ilustración 25. Tabla de verdad para configuración de resolución de motor paso a paso. [Tabla]. 38
Ilustración 26. Muñecos sobre sus carriles. El movimiento de los motores produce el desplazamiento de los muñecos. [Fotografía]. 38
Ilustración 27. Esquema de control de un motor bipolar paso a paso mediante puentes H [Esquema]. (10) 39
Ilustración 28. Implementación de control de motor bipolar paso a paso utilizando dos L293E en paralelo para aumentar la corriente máxima a entregar. [Fotografía] 39
Ilustración 29. Microrruptor D2FD-01L30-1H de OMROM. [Fotografía]. 40
Ilustración 30. Señal eléctrica a través del microrruptor. Antes de llegar a establecerse la señal en 5V, se producen picos con una frecuencia de 1.6kHz. [Fotografía]. 41
Ilustración 31. Señal eléctrica a través del microrruptor. En el flanco de bajada, la señal vuelte a 0V sin rebotes. [Fotografía] 41
Ilustración 32. Tabla de verdad del codificador 8 a 3 SN74LS348N [Tabla]. 42
Ilustración 33. Esquema eléctrico de codificador 8 a 3 [Esquema]. 42
Ilustración 34. Distribución de los microrruptores. [Esquema]. 43
Ilustración 35. Tabla de verdad incluyendo la señal Aaux. [Tabla]. 43
Ilustración 36. Rebotes del microrruptor a la salida del codificador [Fotografía]. 44
Ilustración 37. Flanco de bajada del codificador a la salida del filtro [Fotografía]. 44
Ilustración 38. Esquema de un amplificador operacional como comparador [Esquema]. 45
Ilustración 39. Señal del microrruptor a la salida del comparador [Fotografía]. 45
Ilustración 40. Etapas del procesamiento de señal del subsistema microrruptores [Diagrama]. 46
Ilustración 41. Módulo DFPlayer de DFRobot [Fotografía]. (15) 47
Ilustración 42. Altavoz ABS-230-RD de Pro Signal [Fotografía]. (16) 47
Ilustración 43. Esquema eléctrico del subsistema de audio [Esquema]. 48
Ilustración 44. Solución para reproducir audio mediante microcontroladores STM32 [Esquema]. (17) 48
Ilustración 45. Módulo Receptor AM-HRR30-433 [Fotografía]. 50
Ilustración 46. Decodificador RF600D [Fotografía]. (22) 50

Ilustración 47. Trama de datos que se transmite desde el RF600E al RF600D. [Diagrama]. 51
Ilustración 48. Tabla de código ASCII de la salida serie del decodificador RF600D [Tabla]. 52
Ilustración 49. Esquema eléctrico del subsistema de recepción por radiofrecuencia [Esquema]. 53
Ilustración 50. Placa de desarrollo NUCLEO-F411RE. [Fotografía]. 54
Ilustración 51. Interfaz de configuración de programa STM32CubeMX [Imagen] 55
Ilustración 52. Configuración del reloj del sistema. [Imagen] 55
Ilustración 53.Tabla con todos los pines del Bloque Carriles y su configuración [Tabla]. 56
Ilustración 54. Fuente de alimentación Traco Power [Fotografía]. (16) 57
Ilustración 55. Esquema eléctrico del conexionado del botón MC34231-091-72 [Fotografía]. 57
Ilustración 56. Capas y sus grosores y materiales para PCB de 4 capas y 1.6mm de grosor [Imagen]. (26) 58
Ilustración 57. Detalle de una resistencia de montaje superficial 1206 [Fotografía]. 59
Ilustración 58. Conectores macho y hembra de la serie FASTON 250 de TE Connectivity [Fotografía]. 59
Ilustración 59. Conector CES-119-02-T-D de SAMTEC [Fotografía]. 60
Ilustración 60. Conectores para los servomotores [Fotografía]. 60
Ilustración 61. Conectores MPT 0.572-2,54 de Phoenix Contact [Fotografía]. 60
Ilustración 62. Conector UFL para la antena de los módulos de radiofrecuencia [Fotografía]. 61
Ilustración 63. El ancho de la pista se ajusta al valor asignado mediante la directriz de diseño [Figura]. . 62
Ilustración 64. Led encendido durante proceso de aprendizaje de RF600D [Fotografía]. 62
Ilustración 65. Esquema de los leds de alimentación y resistencias asociadas [Esquema]. 62
Ilustración 66. Montaje del LCD sobre la placa del Bloque Rampa [Fotografía]. 63
Ilustración 67. Fotografía y esquema de los terminales SB10 y SB9 [Imagen]. 63
Ilustración 68. Esquema de rutado de la placa del Bloque Carriles [Esquema]. 64
Ilustración 69. Vista inferior de la placa del Bloque Carriles [Fotografía]. 65
Ilustración 70. Vista superior de la placa del Bloque Carriles [Fotografía]. 65
Ilustración 71. Implementación de StepStick de RepRap sobre placa de prototipado [Fotografía]. 66
Ilustración 72. Fila de leds que se activan mediante pulsación de microrruptor [Fotografía]. 67
Ilustración 73. Convertidor TTL-USB 67
Ilustración 74. Requisitos de la transmisión USART 68
Ilustración 75. Formato de palabra del DFPlayer 68
Ilustración 76. Comandos del DFPlayer impresos en el terminal HERCULES [imagen]. 69
Ilustración 77. Sistema de transmisión por radiofrecuencia sobre placa de prototipado [Fotografía]. 70
Ilustración 78. Trama de datos del receptor de radiofrecuencia RF600E [Imagen]. 70
Ilustración 79. Intervalo de tiempo entre tramas serie del receptor de radiofrecuencia [Fotografía]. 71
Ilustración 80. Tabla de comandos de Radiofrecuencia [Tabla]. 72
Ilustración 81. Diagrama que muestra en qué momento de la partida se envían los comandos RF [Diagrama]. 72
Ilustración 82. Diagrama de flujo del programa principal del Bloque Carriles. Parte 1 [Diagrama]. 73
Ilustración 83. Diagrama de flujo del programa principal del Bloque Carriles. Parte 2 [Diagrama]. 74
Ilustración 84. Diagrama de flujo del programa principal del Bloque Carriles. Parte 3 [Diagrama]. 75
Ilustración 85. Proceso de ensamblaje de las placas de circuito impreso 1 [Fotografía]. 79
Ilustración 86. Proceso de ensamblaje de las placas de circuito impreso 2 [Fotografía]. 79
Ilustración 87. Proceso de ensamblaje de las placas de circuito impreso 3 [Fotografía]. 80
Ilustración 88. Proceso de ensamblaje de las placas de circuito impreso 4 [Fotografía]. 80
Ilustración 89. Proceso de ensamblaje de las placas de circuito impreso 5 [Fotografía]. 81
Ilustración 90. Cable soldado a conector con funda termorretráctil [Fotografía]. 81
Ilustración 91. Conexionado del cableado de la placa de Bloque Rampa [Fotografía]. 82
Ilustración 92. Modificación en el diseño de la PCB. Pata del codificador 8:3 llevada a 5V a través de hilo wrapping [Fotografía]. 84
Ilustración 93. Modificación en el diseño de la PCB. Resistencias de orificio pasante para habilitar el paso de la corriente por ellas [Fotografía]. 85
Ilustración 94. Interfaz de conexión del botón con el microcontrolador (1) [Esquema]. 86
Ilustración 95. Interfaz de conexión del botón con el microcontrolador (2) [Esquema]. 86
Ilustración 96. Conector TS de 3.5mm [Esquema]. (28) 86
Ilustración 97. Modificación en el diseño de la PCB. Resistencias de montaje superficial colocada en paralelo al punto de soldado SB2 [Fotografía]. 87

Índice de Acrónimos.

GDAF: Grupo de Displays y Aplicaciones Fotónicas.

UC3M: Universidad Carlos III de Madrid.

ARASAAC: Sistemas Aumentativos y Alternativos de Comunicación (de Aragón).

PCB: Printed Circuit Board.

TTL: Transistor-Transistor Logic.

USB: Universal Serial Bus.

ARM: Advanced RISC Machine.

RISC: Reduced Instruction Set Computing.

AM: Amplitude Modulation.

FM: Frequency Modulation.

SPDT: Single Pole-Double Throw.

COM: Common.

NO: Normally Open.

NC: Normally Close.

DAC: Digital to Analog Converter.

DMA: Direct Memory Access.

SPI: Serial Periferal Interface.

I2C: Inter-Integrated Circuit.

CRC: Cyclic Redundancy Check.

GPIO: General Purpose Input/Output.

HSI: High Speed Internal.

PLL: Phase Locked Loop.

EXTI: External Interrupt.

USART: Universal Synchronous/Asynchronous Receiver/Transmitter.

TRS: Tip-Ring-Sleeve (Conector).

TS: Tip-Sleeve (Conector).

SB: Solder Bridge.

1. Capítulo 1. Introducción y Objetivos.

1.1. Introducción.

El presente proyecto es fruto de la estrecha colaboración que desde hace varios años mantienen el **Colegio de educación especial San Rafael** en Madrid y el **GDAF-UC3M**, Grupo de Displays y Aplicaciones Fotónicas, del departamento de tecnología electrónica de la **Escuela Politécnica Superior** en Leganés.

Como consecuencia de esta colaboración surgió la oportunidad y la necesidad de realizar un sistema de ayuda para adolescentes con capacidades especiales en su transición hacia la vida adulta (edad comprendida entre los 16 y los 21 años (1)).

Se precisaba por parte del Colegio San Rafael una serie de juegos que sirvieran de estímulo para sus alumnos. Y surgió, entre otras ideas, la de realizar un **juego de carreras** cuyo aspecto y funcionamiento se detallan en las siguientes páginas.

Con este objetivo, dos alumnos de Grado de Ingeniería Electrónica Industrial y Automática, Sergio Aguilera Sevilla y el autor de la presente memoria, acometieron dos Trabajos Fin de Grado que permitieran la realización de uno de dichos juegos. Para ello, ha sido necesaria la aplicación práctica de los conocimientos adquiridos a lo largo de la carrera, así como la necesidad de aprender nuevos conceptos, técnicas y herramientas utilizadas en el ámbito profesional.

1.2. Motivación.

Como se ha mencionado ya, la colaboración entre la universidad y el Colegio San Rafael se lleva produciendo desde hace varios años, y muestra de ello es que el presente proyecto es el decimoquinto (tal y como indica el título del proyecto) que se hace con el objetivo de servir de utilidad al colegio.

El nexo principal de unión entre ambas instituciones es el profesor Ricardo Vergaz Benito, quien puso en contacto a los alumnos autores de este proyecto y colegio para una primera visita en la que varios profesionales del Colegio San Rafael (Javier, Susana y Raquel) mostraron las necesidades demandadas. Enseñaron qué requisitos a nivel funcional debía cumplir el sistema. La gran parte de estos requisitos se recogen en dos conceptos: **Transición a la Vida Adulta** y **Diseño para Todos**.

1.2.1. Transición a la Vida Adulta.

Se define **Transición a la Vida Adulta** como aquel periodo de tiempo entre la finalización del periodo escolar por parte del alumno con discapacidad severa y su completa integración en la sociedad tanto a nivel laboral como a nivel personal. En este periodo de tiempo se buscan desarrollar las siguientes cualidades (1) (2):

- Autonomía personal en la vida diaria.
- Integración social y comunitaria.
- Orientación y formación laboral.

Con este proyecto se pretende potenciar la capacidad de los alumnos a nivel social y comunicativo a través de la competición y el ocio. La intención es que gracias al juego de carreras aumente la interacción social y la creación de lazos emocionales entre ellos.

1.2.2. Diseño para todos.

En el Colegio San Rafael existen alumnos con diferentes capacidades especiales. Por ello, con el objetivo de que cualquiera pueda disfrutar del juego, es importante que el diseño se ciña lo máximo posible a las pautas del "**Diseño para Todos**".

El Diseño para Todos se define, en España según la ley 51/2003, de la siguiente manera:

> *"Diseño para todos: la actividad por la que se concibe o proyecta, desde el origen, y siempre que ello sea posible, entornos, procesos, bienes, productos, servicios, objetos, instrumentos, dispositivos o herramientas, de tal forma que puedan ser utilizados por todas las personas, en la mayor extensión posible."* (3)

Para conseguir el objetivo de que el diseño pueda ser utilizado por "todas las personas, en la mayor extensión posible", se ha intentado cumplir con los principios de Diseño Para Todos:

1. Uso equitativo: el diseño debe ser igualmente atractivo para todos los usuarios, independientemente de las capacidades de estos. Y debe asegurar el mismo nivel de privacidad, garantía y seguridad a todos.

2. Flexibilidad en el uso: se debe facilitar al usuario la correcta interacción con el sistema independientemente del nivel de precisión o exactitud que aplique. Además, el sistema debe adaptarse al ritmo de usuario.

3. Uso simple e intuitivo: el sistema debe estar diseñado tal que su complejidad sea la justa y necesaria para la correcta interpretación de todas las señales por parte del usuario.

4. Información perceptible: la información debe ser transmitida por todos los medios necesarios para que cualquier usuario la pueda interpretar fácilmente, aunque dicha información sea redundante.

5. Tolerancia al error: el diseño del sistema debe introducir técnicas o herramientas para deshacer cualquier acción indeseada, o bien desalentar a acometer dichas acciones que pudieran ser contraproducente para el correcto funcionamiento.

6. Esfuerzo físico bajo: se debe poder interactuar con el sistema aplicando el mínimo esfuerzo posible, y de la forma más natural posible para cada usuario.

7. Tamaño y espacio para su acceso y uso: el tamaño, la posición o el espacio que ocupe el sistema deben ser variables que afecten lo menos posible a la correcta manipulación del mismo. (4)

A lo largo del presente documento se comprobará que el sistema diseñado cumple con todos estos requisitos.

1.2.3. Colegio San Rafael.

El Colegio de Educación Especial San Rafael es un centro especializado en la atención a alumnos con discapacidad motora, intelectual y otros trastornos asociados perteneciente a la **Orden de Hermanos de San Juan de Dios**. Está anexo al Hospital San Rafael, en la Calle Serrano 199, Madrid.

Ilustración 1. Colegio de Educación Especial San Rafael. Fachada [Fotografía]. *(5)*

En él se ofrece una oferta pedagógica que abarca desde la **Educación Infantil** (3-6 años) hasta la aplicación de **programas para la Transición a la Vida Adulta** (16-21 años). Dentro de este último programa es donde se enmarca el presente proyecto.

El colegio garantiza una atención integral a sus alumnos, promoviendo su máximo desarrollo mediante una intervención transdisciplinar a través de la colaboración entre profesionales de distintos ámbitos (profesores, logopedas, fisioterapeutas, enfermeros…). (5)

Ilustración 2. Pictograma Arasaac en acceso a sala Logopedia [Fotografía]. *(5)*

El centro está organizado en salas que, para facilitar a sus alumnos en qué lugar se encuentran, están señalizadas mediante pictogramas *arasaac* como el que se muestra en la ilustración 2. Se tratan d*e* pictogramas que utilizan en el colegio especialmente diseñados para compensar las dificultades de comunicación y lenguaje de personas con discapacidad (6).

Las salas están divididas según el tipo de actividad que se vaya a realizar en ella (sala Snoezelen…) o según la etapa educativa a la que esté orientada. En este caso particular, el proyecto está enfocado a permanecer en la sala asignada al programa de Transición a la Vida Adulta.

1.3. Estado del Arte.

En la actualidad, existen en el mercado productos similares al que se va a diseñar, pero no están pensados para cumplir con las pautas del Diseño para Todos, lo que los hace imposible para su uso por parte de los alumnos del Colegio San Rafael. Además, el coste de un juego de las características como el que se muestra en la ilustración 3 puede ascender a los 20.000 euros (7). El sistema que propusieron desde el colegio es muy parecido a lo que se muestra en la imagen.

Ilustración 3. Juego de Carreras [Imagen]. *(7)*

Aunque existen juegos especialmente diseñados para usuarios con capacidades especiales, por norma estos están orientados a un público infantil.

Según un estudio realizado por AIJU sobre juguetes que existen en el mercado tan solo el 61% han sido valorados como accesibles para personas con discapacidad motora, y solo una quinta parte de ellos son adecuados sin ayuda ni adaptación (ilustración 4).

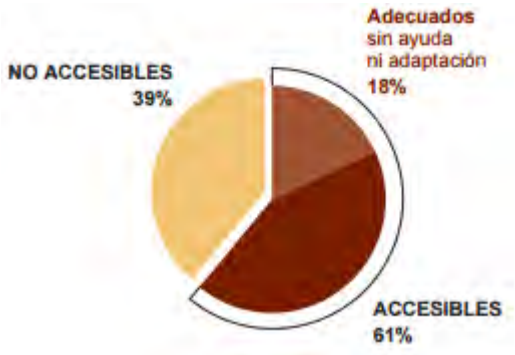


Ilustración 4. El 39% de los juguetes no son adecuados para ser usados por personas con discapacidad motora. [Gráfico]. *(30)*

Esta estadística empeora si se tiene en cuenta además el rango de edad al que va dirigido el producto. En la ilustración 5 solo se observa el porcentaje de productos para niños de hasta 10 años.

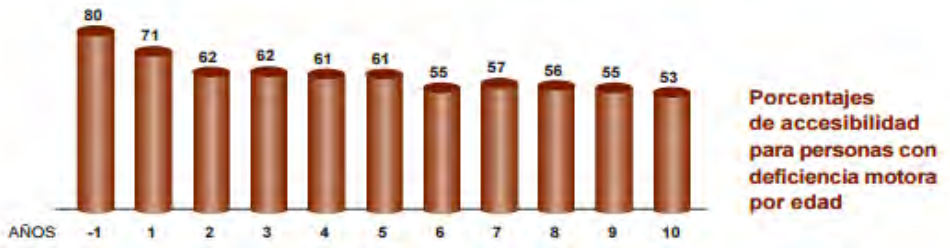


Ilustración 5. A medida que aumenta el rango de edad al que va dirigido el producto, el porcentaje de juguetes adaptados disminuye. [Gráfico]. *(30)*

Por otro lado, en el centro confluyen alumnos con varias discapacidades más allá de discapacidades motoras, y además varios alumnos pueden llegar a padecer varios tipos de discapacidad (visuales, auditivas o motoras). Por tanto, solo optan al 5% de los juguetes existentes en el mercado, que son los que cumplen con los requisitos de "Diseño para Todos" (ilustración 6) (8).

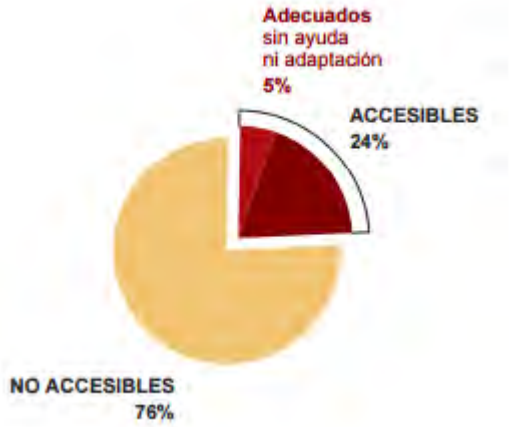


Ilustración 6. Solo el 5% de los productos analizados cumplen con las pautas del "Diseño para Todos". [Gráfico]. *(30)*

Debido a esto, los profesores del centro intentan en la medida de lo posible modificar y adaptar productos comerciales no accesibles a las necesidades de los alumnos. Por ejemplo, tienen pulsadores individuales adaptados a la movilidad de cada alumno, que terminan en un cable Jack macho de dos terminales con los que interactuar con diferentes dispositivos (ilustración 7).

Ilustración 7. De Prado Escudero, M. (2013) Pulsador adaptado para colocar en silla de ruedas que puede ser ajustado en altura y posición. [Fotografía]. *(31)*

En conclusión, la necesidad de este proyecto ya no se basa sólo en la escasez que de por sí tienen estos juegos adaptados, si no en que además esta escasez aumenta según va aumentando la edad a la que están dirigidos.

1.4. Objetivos.

Por todo lo expuesto hasta el momento, el objetivo del presente proyecto (junto con el de Sergio Aguilera Sevilla) consiste en abordar el diseño y construcción de un juego de carreras que, siendo lo más económico posible, cumpla con todas las pautas del Diseño para Todos y que esté enfocado a entretener a los alumnos del programa de Transición a la Vida Adulta, además de ayudarles a aumentar su grado de relación social y física con el entorno.

1.5. Especificaciones del Sistema.

1.5.1. Requisitos del Sistema.

En una primera visita al centro, varios profesionales impusieron una serie de características que el sistema debía de cumplir, muchas de las cuales se guían en los principios del Diseño para Todos. Además, en base a estos mismos principios, se extraen varias pautas básicas que debe seguir el juego:

- La idea principal del funcionamiento del juego que se demanda está basada en los juegos de carreras en los que se lanza una bola hacia distintos agujeros y se mueve el caballo correspondiente al jugador una determinada distancia.

- El sistema debe ir alimentado directamente de la red eléctrica: basándose en experiencias con proyectos anteriores, en el centro a menudo tenían el problema de que la durabilidad de estos se veía limitada debido a un sistema de alimentación por baterías.

- La información debe ser redundante y obligatoriamente transmitida por vía auditiva: de esta forma cumplimos con el principio de *Información Perceptible*. Cualquier usuario será consciente de lo que ocurre en el juego en todo momento. A diferencia de los juegos de carreras clásicos, para que el barrido auditivo sea efectivo es indispensable que el transcurso del juego sea **secuencial**. Es decir, por turnos.

- El juego debe estar pensado para entre 1 y 4 jugadores: antes de empezar la partida, los monitores deben seleccionar el número de jugadores desde el panel de control. Desde este mismo panel de control, y con tal de seguir el principio de *uso equitativo*, se puede seleccionar qué personaje deseará cada jugador, serán intercambiables.

- El diseño del juego debe estar pensado para facilitar la colocación del mismo en cualquier lugar: Por ello se decide dividirlo en dos bloques independientes: un primer bloque donde irán los carriles de los jugadores (Bloque Carriles en adelante), y un segundo bloque consistente en la rampa (Bloque Rampa en adelante). Ambos bloques se comunican mediante radiofrecuencia. Esta idea refuerza el principio de *tamaño y espacio para su acceso y uso*. Está en manos de los monitores colocar el juego donde ellos crean más conveniente para sus alumnos.

- El interfaz único con el jugador debe ser el accionamiento de un pulsador: Dicho pulsador es consistente y fácil de manipular, y va conectado al Bloque Rampa a través de un conector Jack, por lo que puede sustituirse por otro pulsador más conveniente para el usuario. Sobre este principio básico debe basarse el juego. Es decir, el accionamiento del botón debe hacer que a través del movimiento de una bola se mueva el jugador correspondiente, por ello de esta condición se pueden extraer varias características esenciales que debe cumplir el juego:
    - **Esfuerzo físico bajo:** Los alumnos en su mayoría no pueden lanzar una bola, por ello el accionamiento del pulsador hará que una pelota de pingpong caiga desde la parte superior de una rampa.
    - **Aleatoriedad:** La pelota caerá desde la cima de la rampa hasta cinco casillas que determinarán cuánto se moverá el jugador. Debe haber algún mecanismo que haga aleatoria la puntuación en cada accionamiento (uso de servomotores y pivotes de madera colocados sobre la rampa).

  Todo ello implica el cumplimiento de los principios de *esfuerzo físico bajo, flexibilidad en el uso, tolerancia al error y uso simple e intuitivo.*

.

- Dimensiones del Bloque Carriles de 90x60x60 cm y del Bloque Rampa de 50x50x80 cm. Se impusieron unas dimensiones aparentemente grandes pero que resultan lógicas teniendo en cuenta que sobre la rampa debe deslizarse una bola y que los carriles deben ser mínimamente largos para darle una cierta durabilidad al juego.

1.6. Metodología.

Teniendo en cuenta los requisitos que debía tener el sistema, el primer paso consistió en definir conceptualmente qué debía tener cada bloque (microcontroladores, sistema de audio, servomotores, motores paso a paso…).

En segundo lugar, una vez definido cada subsistema, el objetivo fue hacer que la electrónica de cada uno funcionara correctamente. Para aquellos subsistemas que precisaran de programación para comprobar su correcto funcionamiento (como por ejemplo el sistema de audio) se hizo además un pequeño programa aparte.

Definidos todos los bloques y comprobado que funcionaban a nivel individual, el siguiente paso consistió en fabricar los chasis de madera tanto del Bloque Carriles como del Bloque Rampa sobre los que se iba a sustentar toda la electrónica y diseñar e imprimir en plástico las piezas necesarias con la impresora 3D. Al mismo tiempo, para prevenir el tiempo que tardarían en ser fabricadas y llegar las placas de circuito impreso, se empezó con el diseño de las PCB's.

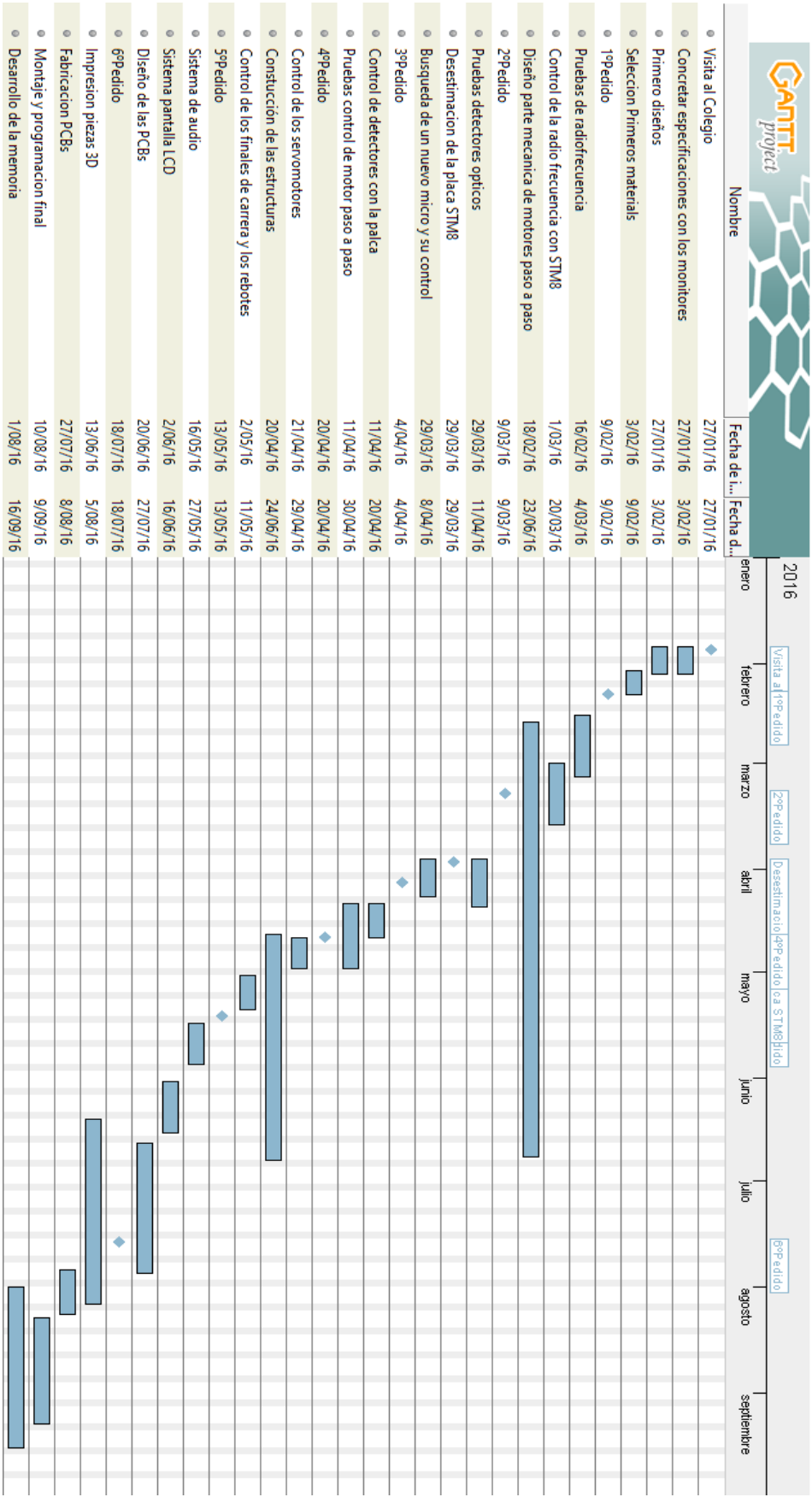
Una vez que hubieron llegado las PCB's se comenzó a soldar todos los componentes para después comenzar con el diseño de un programa para cada bloque que controlara todos los subsistemas. Desde el Bloque Rampa se debía poder establecer la configuración previa (número de jugadores, personajes…) y controlar la caída de la bola en cada una de las casillas, transmitiendo toda la información necesaria al otro bloque. Desde el Bloque Carriles se debía interpretar en todo momento la información que llegaba del Bloque Rampa y actuar en consecuencia (activando motores paso a paso, audio…).

A continuación, con una primera versión de los programas, se comenzó con la integración de la electrónica sobre los chasis mecánicos y la consiguiente fabricación de todos los cables necesarios. Cuando se hubo hecho todo esto, comenzamos una fase de depuración de los programas a base de ensayo y error para hacer correcciones con tal de que el funcionamiento del juego fuera lo mejor posible.

En medio de todo este proceso hay que mencionar que se realizaron varios pedidos de materiales a lo largo de los meses, y que se desestimaron algunos de ellos (como el microcontrolador de 8 bits de STMicroelecronics en favor de uno de 32 bits del mismo fabricante). Además, tras una primera toma de contacto con el colegio para tener claras las especificaciones del proyecto, se mantuvieron contactos según fueron surgiendo dudas con respecto al diseño, para que este fuera lo más óptimo y compatible posible con la filosofía de Diseño para Todos.

Todo el proceso se puede observar en el diagrama de Gantt de siguiente apartado.

1.7. Fases del Proyecto.



1.8. Medios Utilizados.

Para la realización del proyecto se dispuso en todo momento de todas las herramientas disponibles en el laboratorio 1.2.C12 del Departamento de Tecnología Electrónica (ilustración 8):

- Osciloscopios.
- Impresora 3D bq Witbox.
- Fuentes de Alimentación.
- Polímetro
- Componentes electrónicos: resistencias, condensadores, diodos, etc.
- Placas protoboard.
- Otros materiales: cable, herramientas de corte, destornilladores, taladradora, pistola termoplástica…
- Conversor TTL-USB.

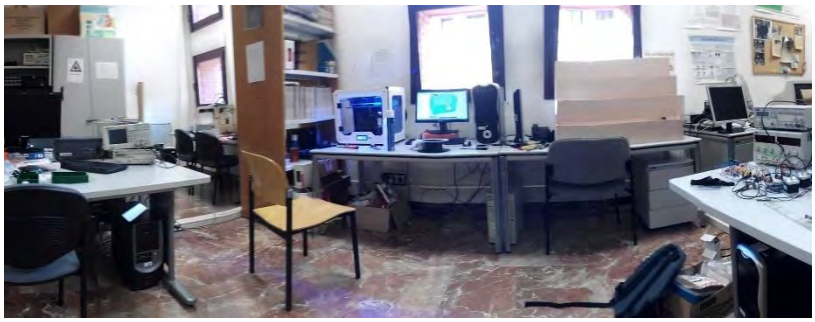


Ilustración 8. Laboratorio 1.2.C12. [Fotografía]

Por otro lado, desde casa también se dispuso de diversos materiales que permitieron avanzar, en el mes de Agosto en especial (ilustración 9):

- Polímetro.
- Fuente de alimentación y osciloscopio[1].
- Puesto de soldadura: soldador JBC de 25W, desoldador, malla de cobre, flux…

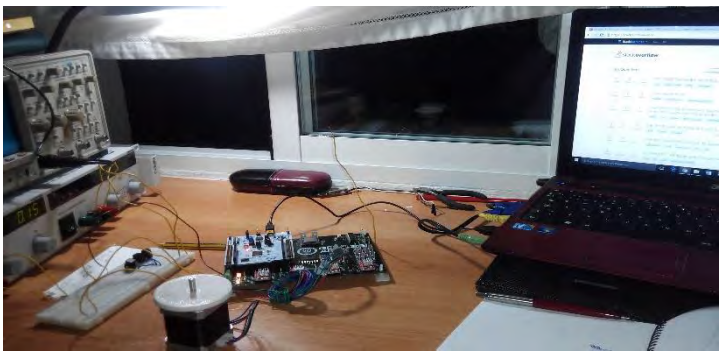


Ilustración 9. Detalle de fuentes de alimentación y osciloscopio, equipo portátil y placa de Bloque Carriles. [Fotografía].

[1] Cortesía de *Indra Sistemas*

En cuanto al software, utilizamos varios programas:

- Altium Designer 2016 – Evaluation License: con este software diseñamos los esquemas eléctricos de los dos bloques, así como las placas de circuito impreso.
- Autocad 2015 Student Version: para los diseños de las piezas 3D.
- STM32CubeMX: herramienta de configuración gráfica que permite generar el código de inicialización en C de cualquier microcontrolador o placa de desarrollo de la familia STM.
- Atollic TrueStudio 5.4.2 Lite Version: IDE (*Integrated Development Tool*) especialmente diseñada para microcontroladores y microprocesadores con arquitectura ARM (*Advanced RISC*[2] *Machine*).
- HERCULES Setup Utility: terminal cuya principal ventaja por la que fue elegido es que es capaz de interpretar los bytes en hexadecimal y mostrártelos de esta forma en tiempo real según van llegando.

1.9. Descripción de la memoria.

La presente memoria tiene partes comunes con el trabajo de Sergio Aguilera Sevilla, ya que ambas forman parte del mismo proyecto. No obstante, a partir del capítulo 1 cada uno se centra en aspectos más concretos. Mientras que Sergio Aguilera se va a centrar en los subsistemas y el programa del Bloque Rampa y la construcción mecánica de los chasis y las piezas 3D, este proyecto se va a centrar en los subsistemas y programa del Bloque Carriles y del proceso de diseño y ensamblaje de las placas de circuito impreso de ambos bloques.

Capítulo 1. Introducción y Objetivos. Este primer bloque es en su mayoría común a los dos trabajos. En él se puede leer una presentación del colegio y la necesidad que lleva a la construcción del proyecto, las fases por las que ha pasado durante su desarrollo y los medios que se utilizaron para llevarlo a cabo. Además de una explicación de los requisitos que deben reunir los dispositivos para personas con capacidades especiales.

Capítulo 2. Diseño del Sistema. Se empieza describiendo el funcionamiento detallado del sistema y un esquema global de ambos bloques para a continuación centrarse en la descripción detallada de todos los subsistemas del Bloque de los Carriles (componentes escogidos, alternativas planteadas y razón de su descarte). Para terminar, se describe el proceso de diseño seguido para las placas de circuito impreso.

Capítulo 3. Implementación del Sistema. Se describen las primeras pruebas realizadas de los subsistemas del Bloque Carriles. Después se continúa con la descripción del firmware del Bloque Carriles y cómo interactúa con el firmware del otro bloque. Se termina con una descripción del proceso de ensamblaje de las PCB's, así como de la fabricación de los cables y colocación de las fuentes y otros elementos sobre los chasis.

Capítulo 4. Pruebas y Resultados Experimentales. Descripción del proceso de depuración del firmware y retoques realizados sobre las placas. Descripción de primeras pruebas de campo.

[2] Reduced Instruction Set Computer: los microprocesadores de este tipo tienen como característica principal que presentan instrucciones de tamaño reducido y fijo. (29)

Capítulo 5. Conclusiones y Posibles Líneas Futuras. En esta parte se hará un repaso a los objetivos marcados y las conclusiones personales que deja este proyecto. También se hablará de posibles mejoras que se pueden realizar. Y se finalizará con el presupuesto, que será común y dará una idea del coste total que tiene el proyecto.

2. Capítulo 2. Diseño del Sistema.

2.1. Funcionamiento detallado del juego.

A fin de facilitar la comprensión, se avanzan aquí algunas fotografías del sistema final para describir lo más visualmente posible el funcionamiento del juego. No obstante, en los apartados posteriores se describirá cómo se llegó al diseño de toda esta implementación.

El juego comenzará con la elección del número de jugadores que va a haber en la partida (entre uno y cuatro) desde el panel de control, situado en un lateral de rampa. A continuación, desde este mismo panel, se seleccionaría qué personaje va a desear ser cada uno de los jugadores, teniendo como opción ser un dinosaurio, un coche, una moto o un caballo (ilustración 10).

Ilustración 10. Mensaje en LCD: "Seleccione número de jugadores." [Fotografía].

Una vez se seleccione el número de jugadores y sus personajes, se activan los servomotores y aparecería en la pantalla del panel de control un mensaje:

"Jugador 1:

Pulse botón Rojo"

El juego en este momento quedará a la espera de la pulsación del botón rojo (o de otro pulsador que hubiera sido previamente conectado). Cuando se pulse, la compuerta que está en la parte de arriba de la rampa se abrirá, al mismo tiempo que los tres servomotores de la rampa comenzarán a moverse de forma aleatoria (ilustración 11).

Ilustración 11. Bloque Rampa. La bola cae de la caseta de la parte superior. [Fotografía].

La pelota de pingpong, que estaba en la caseta, comenzará a descender a través de unos pivotes distribuidos tal que forman una máquina de Galton[3], después atravesará las paletas movidas por los servomotores de la rampa para por último caer en una de las cinco casillas que hay en la parte inferior de la rampa distribuidas a lo ancho de esta (ilustración 12).

En función de donde caiga la pelota del jugador que en ese momento hubiera tirado se sumará una determinada puntuación, siendo 20 puntos la casilla del centro, 15 puntos las casillas colindantes con la casilla central, y 10 puntos las casillas de los extremos.

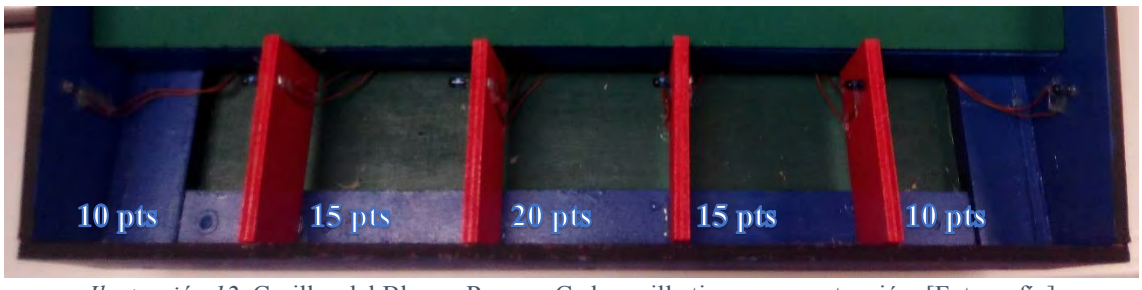


Ilustración 12. Casillas del Bloque Rampa. Cada casilla tiene una puntuación. [Fotografía]

[3] La máquina de Galton hace que la distribución de las casillas en las que cae la bola a lo largo de las partidas sea binomial (33).

El motor correspondiente al jugador que haya tirado se moverá de tal manera que el muñeco asociado se desplace tantos centímetros como puntos se consiga. Es decir, si se ha caído en la casilla central y se han conseguido 20 puntos, el carril se moverá 20 centímetros. La longitud de polea a polea en un mismo carril es de 70 centímetros (ilustración 13), por lo que es lógico pensar que la puntuación para ganar la partida sea de 70 puntos.

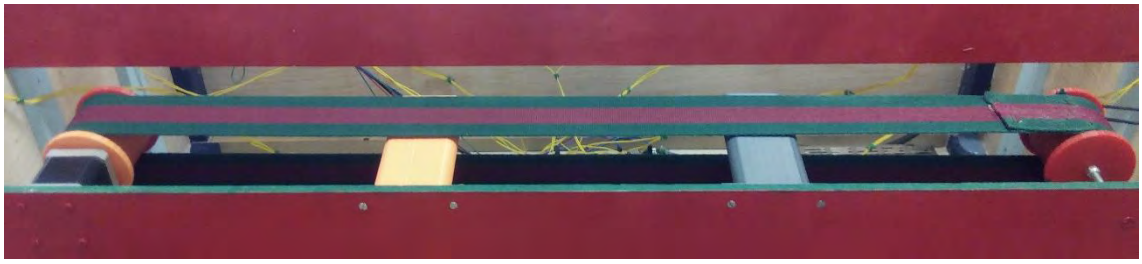


Ilustración 13. Un carril del Bloque Carriles. Su longitud es de 70 cm. [Fotografía]

Cuando termine de desplazarse el carril del jugador que acaba de terminar su turno, es necesario llevar la bola de nuevo a la caseta, lo cual hará manualmente un monitor o eventualmente, si tiene movilidad, un alumno. En este punto es posible que el siguiente jugador necesite usar otro pulsador. Por tanto, antes de que se habilite la opción de pulsar el botón para abrir la compuerta de la caseta se tendrá que presionar el botón OK (ilustración 14). Hasta que no se pulse ese botón, no es posible continuar con la partida. Esto se hace para dar tiempo de colocar la bola de nuevo en la caseta por parte del monitor, y para cambiar de pulsador si fuera necesario.

Ilustración 14. Mensaje en LCD: "Pulse OK para continuar." [Fotografía].

Una vez pulsado el botón OK, se esperará la pulsación del botón para el accionamiento de servomotores de la rampa y apertura de compuerta.

Se repetirá todo el proceso de nuevo para sucesivos jugadores, y se seguirá repitiendo durante tantas rondas como sea necesario hasta que uno de los jugadores alcance los 70 puntos.

Si en una misma ronda varios jugadores superan o igualan los 70 puntos, ganará aquel jugador que tenga más puntos. Si dos o más jugadores tienen la misma puntuación,

ganará el jugador que pulsó el botón en último lugar. En la ilustración 15 se muestra un ejemplo de mensaje de victoria por parte del jugador 1 con 85 puntos.

Ilustración 15. Mensaje en LCD: "Jugador 1 gana con 85 puntos." [Fotografía].

Para volver a empezar una nueva partida, basta con pulsar el botón reset (ilustración 16).

Ilustración 16. Mensaje en LCD: "Pulse RESET para nueva partida." [Fotografía}.

Durante todo el transcurso de la partida se produce un barrido auditivo de todo lo que ocurre (elección del número de jugadores, elección de personajes, puntuación de cada jugador en cada momento, mensaje de victoria, canción de victoria etc.).

2.2. Diagrama de bloques del sistema.

El juego de carreras está claramente dividido en dos bloques (ilustraciones 17 y 20):

- Bloque Rampa

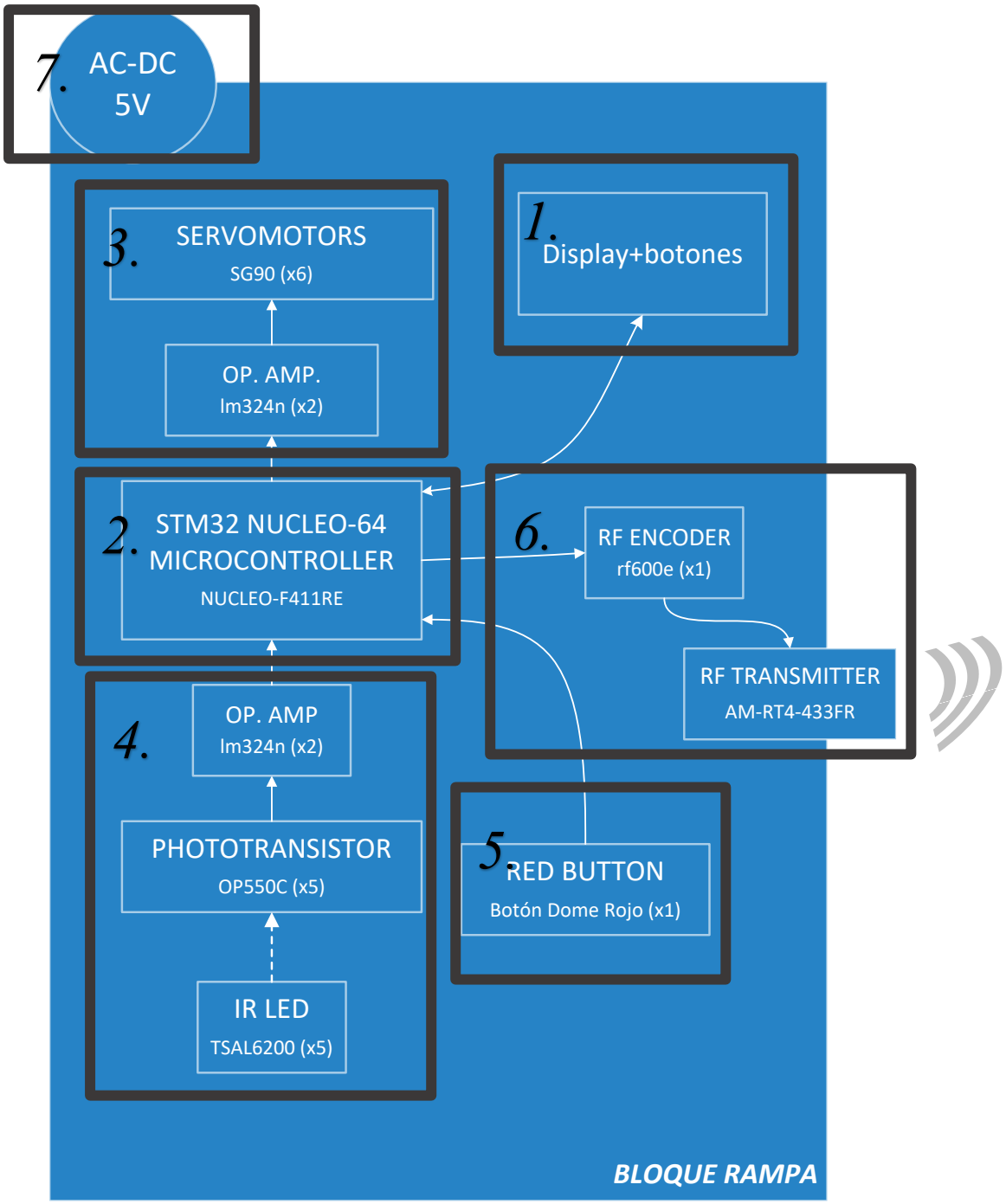


Ilustración 17. Diagrama de bloques del Bloque Rampa. [Diagrama].

1. El panel de control, entendiendo éste como el conjunto LCD+botones (ilustración 18). Mantiene una bidireccionalidad con el "núcleo" del Bloque Rampa. A través de los botones se establece el número de jugadores, se eligen personajes y se controla el juego, y mediante el LCD se transmite de forma visual la información necesaria para ver en qué punto del juego se está.

Ilustración 18. Panel de control, [Fotografía].

2. El núcleo del Bloque Rampa es el que recoge todas las señales y las interpreta de forma adecuada. Genera, en función de las señales recibidas, otras señales para el correcto funcionamiento del sistema. Está gobernado por un microcontrolador de 32 bits del que se hablará con más detalle en el apartado 2.3.5.
3. Los servomotores. Hay cinco. Dos sirven para subir y bajar la trampilla de la caseta. Los otros tres están sobre la rampa y son los encargados de mover las paletas que hay en ella. Forman parte de un mismo bloque porque la electrónica para controlar todos ellos es común.
4. Sensores infrarrojos. Hay cinco, uno en cada casilla. Los sensores están compuestos por un led infrarrojo y un fototransistor. Cuando se corta el haz de luz por el paso de la bola, se envía una señal eléctrica al microcontrolador.
5. Botón Rojo. Botón que por defecto se utiliza para accionamiento de la trampilla (ilustración 19).

Ilustración 19. Botón Dome Rojo [Fotografía]. *(9)*

6. Transmisor por radiofrecuencia: Es capaz de recibir hasta 4 señales simultáneamente y codificarlas. Con estas cuatro señales se mandan los comandos necesarios al Bloque Carriles.

7. Alimentación: Todo el Bloque Rampa se alimenta con una tensión de 5V (o 3,3V, tensión que se extrae de la placa del microcontrolador).

- Bloque Carriles

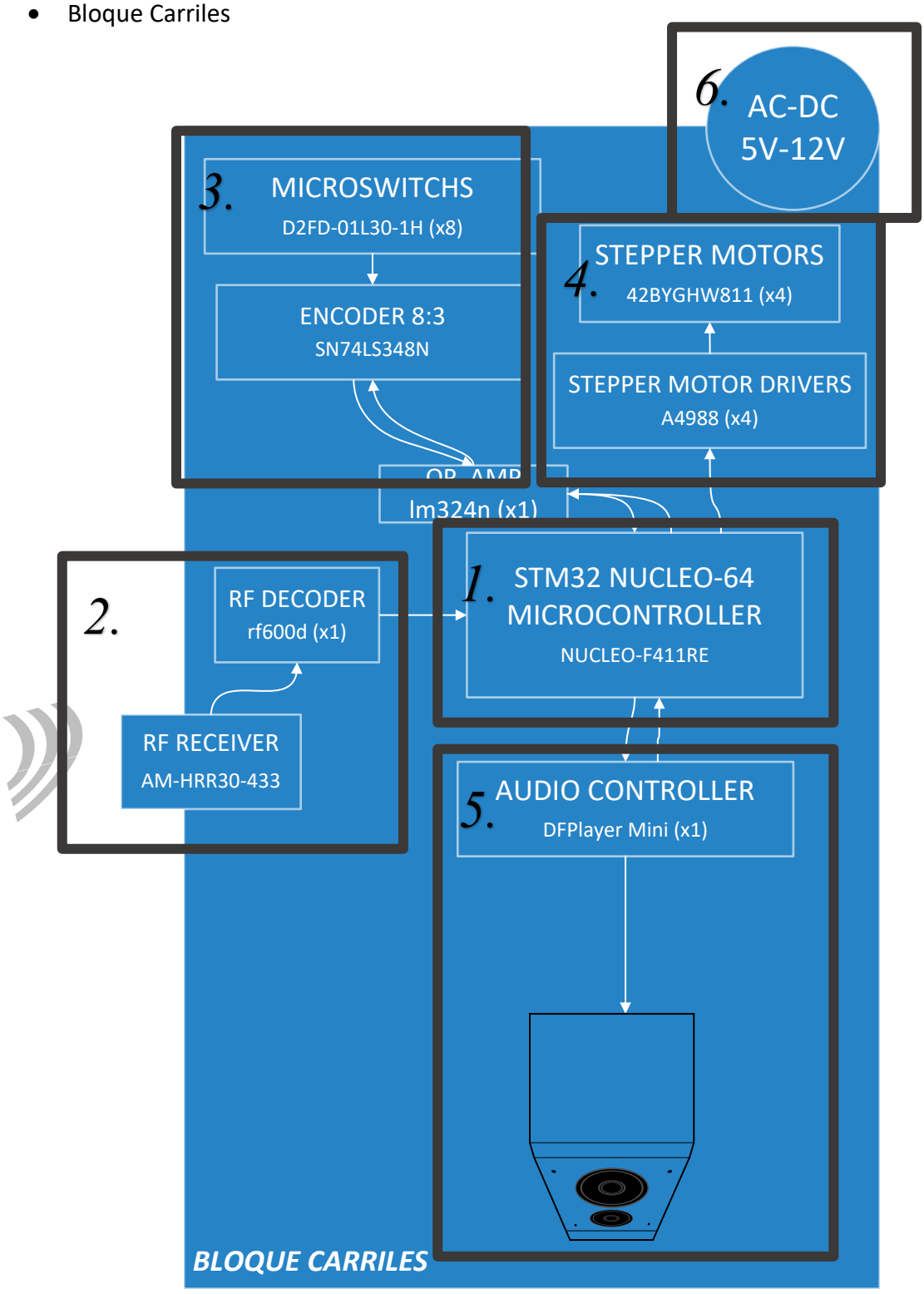


Ilustración 20. Diagrama de bloques del Bloque Carriles [Diagrama].

1. El núcleo del Bloque Carriles es el que recoge todas las señales y las interpreta de forma adecuada. Genera, en función de las señales recibidas, otras señales para el correcto funcionamiento del sistema. Está gobernado por un microcontrolador de 32 bits del que se hablará con más detalle en el apartado 2.3.5.
2. Receptor por radiofrecuencia. Formado por dos bloques, un receptor de radiofrecuencia configurable para trabajar en AM o en FM (en este caso AM) y un decodificador que permite la lectura de la señal en serie o en paralelo (trabajamos en serie).
3. Microinterruptores. Hay ocho, dos por carril. Se sitúa uno en cada extremo del carril con tal de que en caso de que uno de los muñecos llegue al extremo, choque con los sensores finales de carrera y se mande la señal eléctrica correspondiente para que se pare el motor.
4. Motores paso a paso. Hay 4, son los encargados de mover los muñecos sobre la cinta.
5. Sistema de audio. Formado por un integrado capaz de reproducir audios desde una tarjeta microSD y dos altavoces conectados en paralelo.
6. Alimentación. De 5V para todo el sistema y 12V para los motores paso a paso.

2.3. Bloque Carriles.

2.3.1. Motores paso a paso.

Descripción

El elemento utilizado para ejecutar la acción mecánica de desplazar el muñeco a lo largo de un carril es el motor paso a paso. Se trata del motor **42BYGHW811** (ilustración 21), muy utilizado en impresoras 3D.

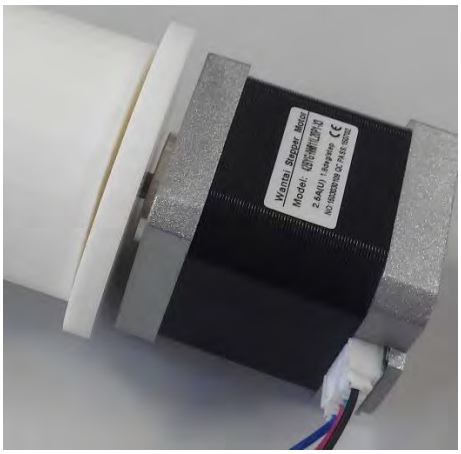


Ilustración 21. Motor bipolar paso a paso 42BYGHW811. [Fotografía]

Como principales características tiene:

- Resolución de paso: 1.8 grados.
- Resistencia de fase: 1.25 ohm.
- Inductancia de fase: 1.8 mH.
- Torque en bloqueo (con devanados energizados): 0.471Nm
- Torque en reposo (con devanados de-energizados): 0.027Nm

- Tamaño: NEMA17
- Corriente máxima: 2.5A

Se trata de un **motor bipolar**, y como tal tiene dos fases y un devanado por fase. Se puede acceder a cada uno de los extremos de los devanados por lo que tiene 4 pines. En un motor bipolar, en función de la corriente de polarización que se aplique a cada una de las fases en cada momento, es posible hacer que el motor se mueva hacia la izquierda, hacia la derecha, o bloquearlo.

El par del motor es proporcional a la intensidad de corriente que se aplica en las bobinas. A mayor corriente, mayor es el par.

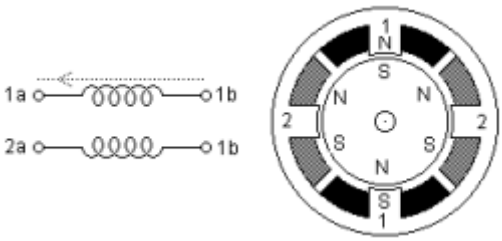


Ilustración 22. Motor bipolar. [imagen]. *(10)*

Se puede cambiar la dirección en que fluye la corriente en cada uno de los devanados invirtiendo la tensión de alimentación. Si por ejemplo aplicáramos una tensión positiva sentido 1b-2a (devanado 1 -parte superior e inferior- en ilustración 22) entonces provocaríamos un desplazamiento de 1 paso. La resolución del paso viene determinada por el número de polos que tiene el motor, en el caso de la ilustración 22 se observa que el rotor tiene 6 polos (360º/6 = 60º de resolución).

Siguiendo con el ejemplo anterior, si además de energizar el devanado 1 en sentido positivo se energiza el devanado 2 en sentido negativo, se tiene que el movimiento es igual, pero maximizando el torque del motor. Si por el contrario este segundo devanado se energiza en sentido positivo, el motor queda bloqueado.

Por otro lado, se puede aumentar la resolución de un paso alternando entre polarizar solo un devanado en una secuencia, y alternar los dos a la siguiente.

El driver utilizado para controlar cada uno de los motores paso a paso es el **StepStick** de RepRap (ilustración 23), basado en el driver de **A4988 Stepper Motor Driver Carrier** de Pololu.

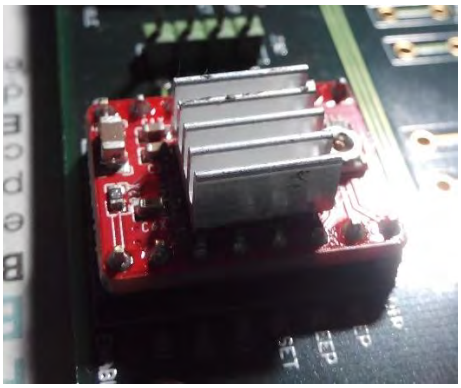


Ilustración 23. Driver A4988 [Fotografía]

Este driver tiene el integrado **A4988** de *Allegro Mycrosystems* embebido. Se trata de un integrado especialmente diseñado para facilitar el control de los motores paso a paso bipolares. Es capaz de trabajar con estos motores con una resolución de hasta la dieciseisava parte de un paso. Tiene capacidad de entregar hasta 2A. Como característica clave para el desarrollo de este subsistema tiene que, aplicando un simple pulso al terminal STEP es suficiente para mover el motor un paso, lo que a nivel de programación facilita mucho el desarrollo.

El driver posee un potenciómetro a partir del cual se puede limitar la corriente que se entregar al motor. Esta corriente viene dada por la relación:

$$I = \frac{Vref}{8 \cdot Rs} \quad (11)$$

Siendo *Vref* la tensión de referencia que se utiliza para ajustar la corriente y Rs la resistencia de sensado interna del driver, que es constante e igual a 0,2ohm.

En fase de pruebas se observó que la corriente óptima necesaria para que el motor pudiera mover la correa de forma eficiente y sin llegar a saturarse estaba en torno a los 0.5A, por ello se han ajustado cada uno de los cuatro potenciómetros para que *Vref* fuera igual a 8·0,2ohm·0,5A=0,8V.

El *pinout* del driver se muestra en la ilustración 24:

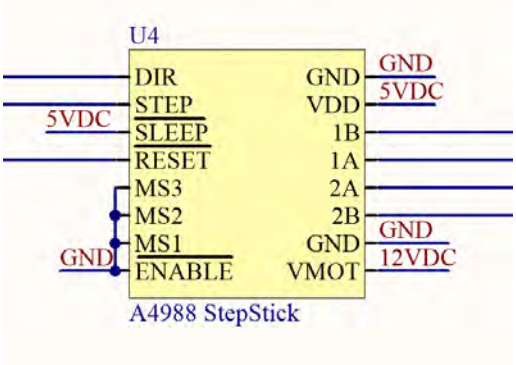


Ilustración 24. Esquemático del driver StepStick de RepRap [Esquema].

- Vdd: Tensión de la lógica del integrado. La máxima que puede soportar es de 5,5V (12), por lo que se pone a 5V al igual que la mayor parte de la lógica del Bloque Carriles.

- Vmot: Tensión de trabajo del driver para mover los motores paso a paso. Se aplican 12V.

- MS3, MS2 y MS1: mediante estos pines se establece qué resolución se desea aplicar al motor, según tabla de ilustración 25.

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Ilustración 25. Tabla de verdad para configuración de resolución de motor paso a paso. [Tabla].

En este caso los tres pines están llevados a tierra para una resolución de un paso.

- STEP: Aplicando un pulso positivo a este pin el motor se mueve un paso. Señal proveniente del microcontrolador.
- DIR: Un cambio en el nivel lógico de este pin produce un cambio en el sentido de giro del motor. Señal proveniente del microcontrolador.
- RESET: A nivel bajo desactiva todas las salidas de los transistores. Hasta que no se activa se ignora cualquier acción que se aplique sobre STEP. Señal proveniente del microcontrolador.
- SLEEP: Señal que cuando está a nivel bajo desactiva la mayor parte de la circuitería interna del A4988. Un nivel alto en este pin hace que el integrado trabaje un modo de funcionamiento normal. Se lleva directamente a 5V en esta aplicación por no ser de especial interés su uso.
- ENABLE: Entrada cuya función es activar o desactivar la salida de los transistores. Se mantiene a tierra en esta aplicación para mantener transistores activados.
- 1B, 1A, 2B, 2A: Son las salidas que van a las fases de los devanados. Se conectan una a una directamente al motor paso a paso.

Por tanto, mediante cuatro motores y sus correspondientes cuatro drivers, se controla el movimiento de los cuatro muñecos sobre cada una de sus correas (ilustración 26). Los aspectos mecánicos se detallan en el proyecto de Sergio Aguilera Sevilla.

Ilustración 26. Muñecos sobre sus carriles. El movimiento de los motores produce el desplazamiento de los muñecos. [Fotografía].

Tanto los motores paso a paso como el driver descrito son ampliamente utilizados en impresoras 3D. Por la fiabilidad demostrada en su uso, la facilidad de encontrarlos en el mercado y la simplicidad de su implementación, se ha optado por esta solución.

Alternativa de control

En un primer momento, la idea que se planteó fue la de controlar los motores paso a paso a través de un **L293E**, integrado con 4 salidas push-pull, 2 para cada devanado de un motor bipolar paso a paso.

Cada fase por tanto se controla mediante 4 transistores (lo que se denomina un puente H) por lo que mediante la activación o desactivación de cada uno de ellos se puede polarizar el devanado en un sentido u otro. En la ilustración 27 se puede ver que, si los transistores Q1 y Q4 se activan mientras que Q2 y Q3 están apagados, la corriente fluirá de izquierda a derecha del devanado.

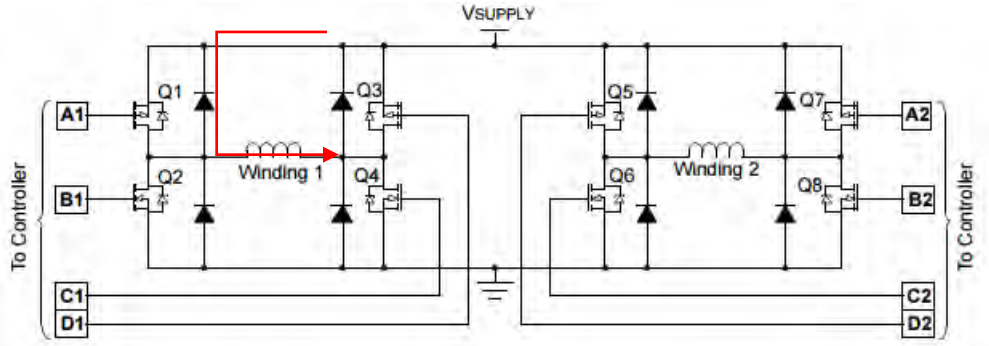


Ilustración 27. Esquema de control de un motor bipolar paso a paso mediante puentes H [Esquema]. *(10)*

Aunque finalmente se logró implementar esta solución para controlar el motor paso a paso (Ilustración 28), fue descartada por varias razones:

- El riesgo de alterar el orden en que los transistores deben abrirse para polarizar los devanados de forma correcta es alto.
- La programación se complica y resulta más difícil de implementar que utilizando el A4988. Habría que generar una secuencia correcta de apertura de transistores para cada sentido de giro o resolución deseada.
- En lugar de utilizar 12 pines para el control de los motores habría que utilizar 16.
- El riesgo de fallos a la hora de implementar esta solución sobre una PCB es mayor que con el StepStick de RepRap, ya que este último viene en un módulo con toda la electrónica auxiliar necesaria para el A4988.

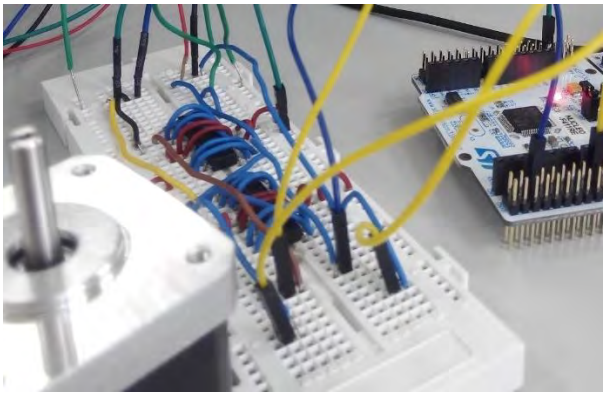


Ilustración 28. Implementación de control de motor bipolar paso a paso utilizando dos L293E en paralelo para aumentar la corriente máxima a entregar. [Fotografía]

En los anexos se puede acceder a los esquemáticos completos de los motores paso a paso y sus drivers. Y en el apartado anterior su descripción a nivel global.

2.3.2. Microrruptores.

Descripción

Los microrruptores hacen la función de elemento de seguridad ante la posibilidad de que los muñecos lleguen a uno de los extremos del carril y continúen su movimiento debidamente.

Aunque a nivel de programación se ha intentado evitar que esto ocurra (cuidando que cuando se llegue a 70 puntos el motor no gire más), resulta inevitable que en mitad de una partida si se tienen 65 puntos se puedan conseguir 20 más. Se trata por tanto de dar un nivel redundante de seguridad al sistema para evitar su desgaste todo lo posible. Se necesitan ocho microrruptores, uno para cada extremo de cada uno de los cuatro carriles.

El sensor final de carrera (que en definitiva es eso) es el **D2FD-01L30-1H** de OMROM (ilustración 29).

Ilustración 29. Microrruptor D2FD-01L30-1H de OMROM. [Fotografía].

Tiene una configuración SPDT[4] con tres pines: NC[4], NO[4] y COM[4]. La idea principal es que el accionamiento del microrruptor provoque la alteración de una señal eléctrica hacia el microcontrolador para interpretarla.

En las ilustraciones 30 y 31 se muestra una señal de 5V al pasar a través del microrruptor y lo que le ocurre cuando este se cierra y se abre. El terminal COM es el que

[4] SPDT: Single Pole – Double Throw
NO: Normally Open
NC: Normally Close
COM: Common

posteriormente va a la electrónica que va a tratar esta señal antes de llegar al microcontrolador, el NC está a un nivel de tensión y NO a otro. Cuando se produce una pulsación la señal pasa de 5V a 0V de forma abrupta y sin rebotes, que es lo ideal. Cuando se suelta el pulsador la señal pasa de 0V a 5V de forma discontinua y con muchos picos. Estos rebotes alcanzan frecuencias de hasta 1,6kHz.

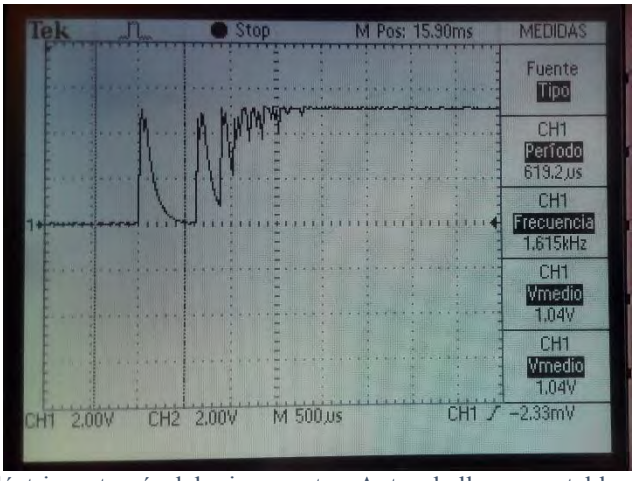


Ilustración 30. Señal eléctrica a través del microrruptor. Antes de llegar a establecerse la señal en 5V, se producen picos con una frecuencia de 1.6kHz. [Fotografía].

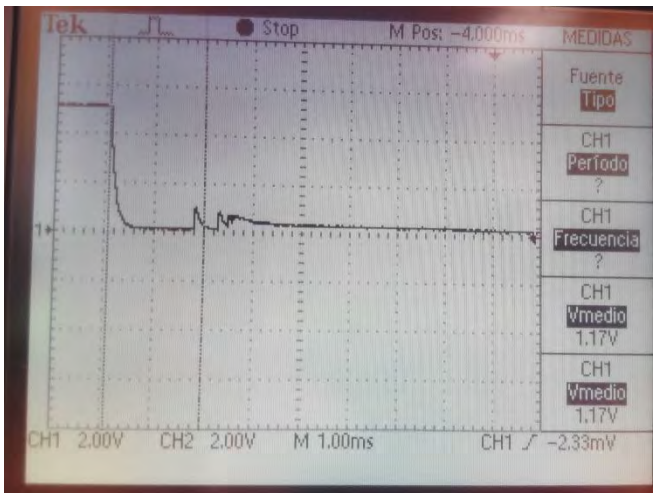


Ilustración 31. Señal eléctrica a través del microrruptor. En el flanco de bajada, la señal vuelte a 0V sin rebotes. [Fotografía]

Si la señal se comportara en cualquiera de las dos situaciones como en la ilustración 32, esta se podría llevar (previa reducción de tensión a unos niveles que soporte el micro -3,3V-) a uno de los pines del microcontrolador e interpretarla. Sin embargo, debido a los rebotes que se producen en el flanco de subida, hay que tratar la señal antes de que llegue al microcontrolador. Para ello se diseña un simple filtro paso bajo teniendo en cuenta la frecuencia de corte y la ganancia deseada.

Antes de continuar con el diseño del filtro conviene pensar que, si se aplica la solución de microrruptor mas filtro, habría que implementar ocho filtros para ocho señales diferentes y ocupar ocho pines del microcontrolador. Por ello, con el objetivo de ahorrar recursos, se diseña una etapa intermedia mediante la cual se busca reducir el número de señales que sea necesario interpretar por parte de microcontrolador.

El integrado **SN74LS348N** es un codificador 8:3 de Texas Instruments mediante el cual logramos reducir el número de señales. De la hoja de datos del integrado se extrae la tabla de verdad (ilustración 32):

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	Z	Z	Z	H	H
L	H	H	H	H	H	H	H	H	Z	Z	Z	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

Ilustración 32. Tabla de verdad del codificador 8 a 3 SN74LS348N [Tabla].

Con el pin EI (Enable Input) a nivel lógico alto, la lectura de las entradas del codificador es ignorada y las salidas A2, A1 y A0 están en alta impedancia. Cuando EI está a nivel bajo, la lectura de las salidas es efectuada. La entrada 7 es la de mayor peso, cuando se produce un cambio a nivel bajo en esta entrada, todas las demás son ignoradas y las salidas se ponen a nivel bajo. A medida que se activan a nivel bajo las entradas 6, 5, 4… del codificador, las salidas siguen una secuencia binaria siendo A2 el bit más significativo y A0 el que menos.

Cuando se activa el bit 7 las salidas se ponen a 0, que es el estado inicial. Por tanto, con 3 bits podemos ser capaces de leer hasta 7 pulsaciones de microrruptores diferentes en función de las entradas. Necesitamos añadir una línea extra para interpretar el estado del microrruptor faltante y mantener a nivel alto la entrada 7 para que su estado no se imponga sobre el resto de entradas, por tanto habrá un microrruptor que no irá al codificador, si no directamente hacia uno de los filtros.

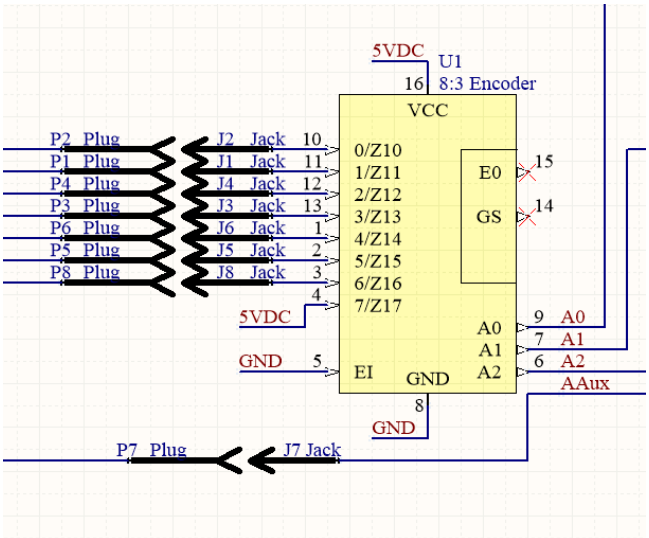


Ilustración 33. Esquema eléctrico de codificador 8 a 3 [Esquema].

En la ilustración 33 se observa la presencia de esta línea extra. Una pulsación del microrruptor supone un cambio a nivel bajo, este cambio a nivel bajo resulta sin embargo en un cambio a nivel alto en las salidas de codificador. Por tanto, para que en el microrruptor que no va al codificador la pulsación del mismo se interprete con un flanco de subida, se conecta con el NO a 5V y el NC a GND (al contrario que el resto de microrruptores).

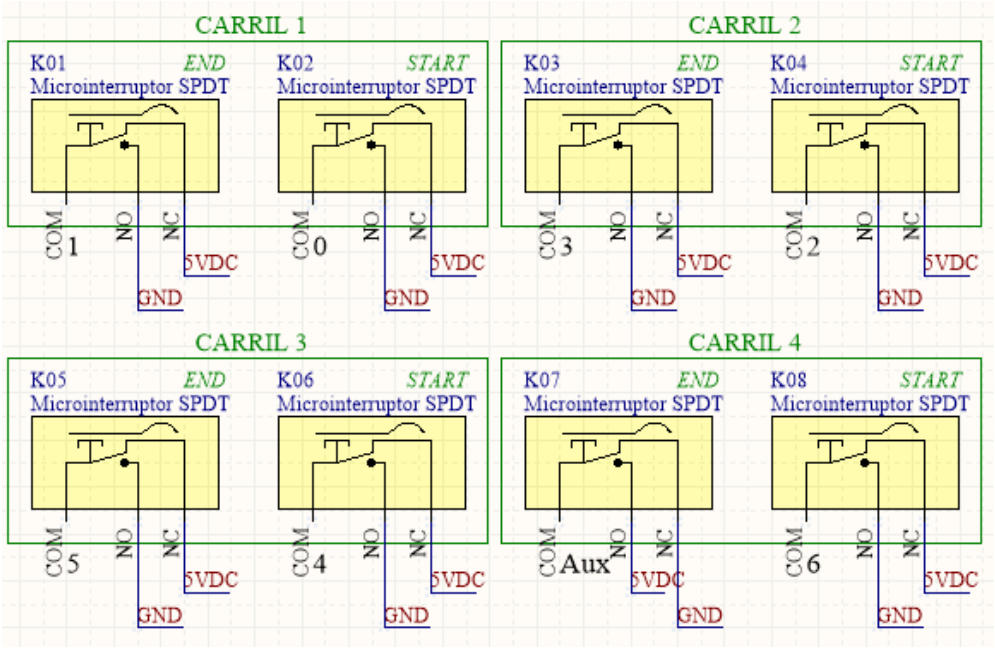


Ilustración 34. Distribución de los microrruptores. [Esquema].

Tenemos por tanto los microrruptores distribuidos en cuatro carriles, con un microrruptor en cada extremo (ilustración 34). Y conectados de forma que la tabla de verdad definitiva, incluyendo al microrruptor auxiliar, es la que se muestra en la ilustración 35.

	EI	0	1	2	3	4	5	6	7	A2	A1	A0	Aaux
	1	X	X	X	X	X	X	X	X	Z	Z	Z	-
	0	1	1	1	1	1	1	1	1	Z	Z	Z	-
	0	X	X	X	X	X	X	X	0	0	0	0	-
K08	0	X	X	X	X	X	X	0	1	0	0	1	-
K05	0	X	X	X	X	X	0	1	1	0	1	0	-
K06	0	X	X	X	X	0	1	1	1	0	1	1	-
K03	0	X	X	X	0	1	1	1	1	1	0	0	-
K04	0	X	X	0	1	1	1	1	1	1	0	1	-
K01	0	X	0	1	1	1	1	1	1	1	1	0	-
K02	0	0	1	1	1	1	1	1	1	1	1	1	-
K07	X	X	X	X	X	X	X	X	X	-	-	-	1

Ilustración 35. Tabla de verdad incluyendo la señal Aaux. [Tabla].

Llegados a este punto, la señal que sale del codificador tiene la forma que se muestra en la ilustración 36. Por ello, tal y como se ha comentado anteriormente, es necesario el diseño de un filtro paso bajo que suavice los rebotes que se producen.

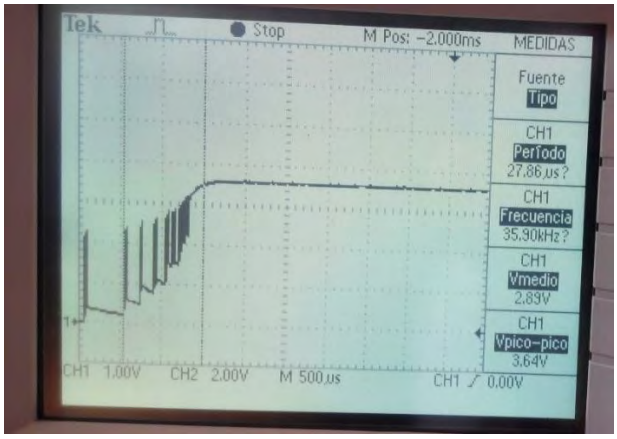


Ilustración 36. Rebotes del microrruptor a la salida del codificador [Fotografía].

Como consecuencia del retardo de propagación de la señal del codificador, que es de entre 11ns y 30ns (según datasheet (13)) la frecuencia de los rebotes ya no es de 1,6kHz como en el caso anterior, si no de 36kHz (ilustración 36), no obstante diseñaremos el filtro con una frecuencia de corte igual a 1,6kHz para asegurar eliminar todos estos rebotes.

Sea la frecuencia de corte:

$$f_c = \frac{1}{2\pi RC} = 1{,}6kHz$$

Y la ganancia (en continua):

$$|G| = \frac{\frac{1}{C}}{\sqrt{R^2 + \left(\frac{1}{C}\right)^2}} = 1$$

Se tiene que R es igual a 100 ohm y el valor de C igual a 1uF. Con este filtro, la señal a la salida del mismo es la que se muestra en la figura 37:

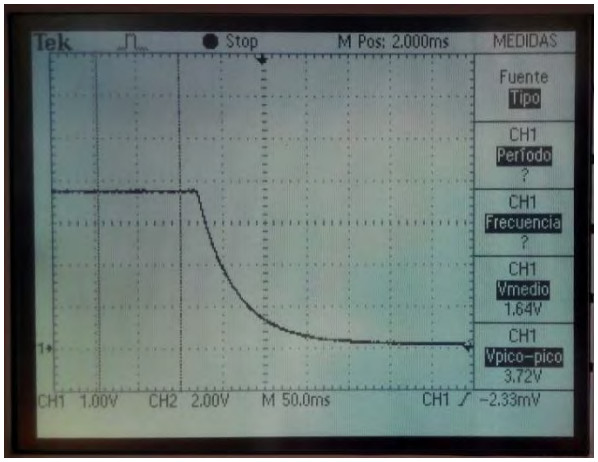


Ilustración 37. Flanco de bajada del codificador a la salida del filtro [Fotografía].

Por último, antes de llevar la señal al microcontrolador, hay que reducir el nivel de tensión que es de 5V a un nivel que pueda ser interpretado (de 3,3V aproximadamente). Se hace necesaria una última etapa, que va a consistir en un amplificador operacional en modo comparador, concretamente el **LM324N** de Texas Instruments.

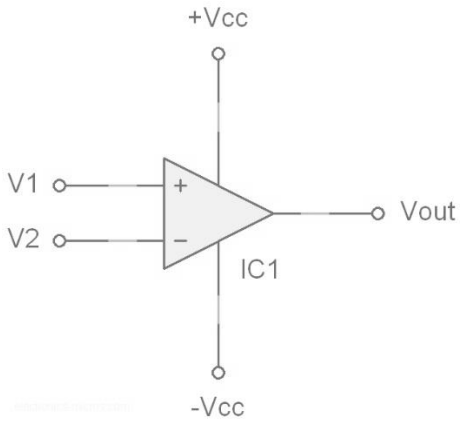


Ilustración 38. Esquema de un amplificador operacional como comparador [Esquema].

Siguiendo el esquema de la ilustración, +Vcc y V2 están a 3,3V, tensión que se saca de la placa de desarrollo del microcontrolador. -Vcc va a tierra y V1 va a la salida del filtro.

Como cualquier comparador, cuando V1 es mayor que V2, Vout se pone a +Vcc (3,3V) en este caso. Cuando V1 es menor que V2, Vout se pone a -Vcc (GND). Al final, la señal que llega al microcontrolador es la que se muestra en la ilustración 39, el nivel de tensión es de 2,12V en lugar de 3.3V debido al output swing del amplificador, pero no resulta un problema dado que este nivel de tensión ya es detectado por el microcontrolador.

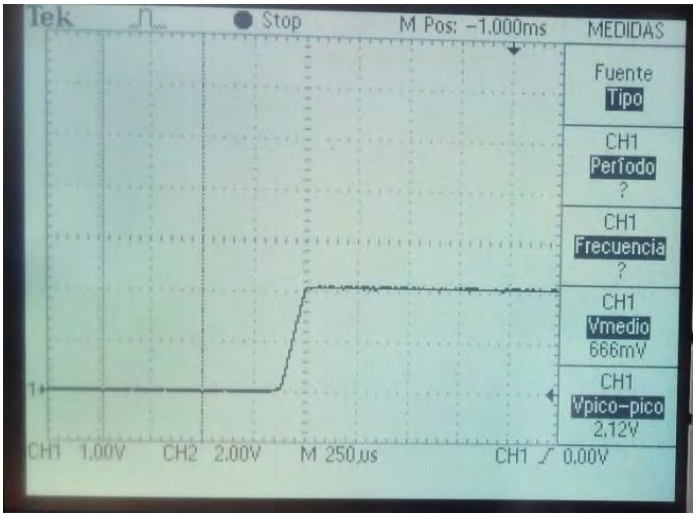


Ilustración 39. Señal del microrruptor a la salida del comparador [Fotografía].

Como último detalle, resulta obligatorio poner una resistencia de descarga hacia tierra entre el filtro y la entrada del amplificador operacional. De lo contrario, cuando la señal pasa de 5V a 0V, esa transición no se produce en la salida del filtro porque la corriente no tiene un camino por el que fluir. Esto se debe a la influencia de la

impedancia de entrada del LM324 (14), que viene dada por la tensión de entrada y la corriente de polarización máxima a la entrada:

$$Rin = \frac{V_{in}}{I_{bias}} = \frac{5V}{100nA} = 50Mohm$$

Por ello se pone una resistencia de 1kohm entre la salida de cada filtro y cada entrada del amplificador.

Así todo, para implementar este subsistema se necesita:

- 8 microrruptores.
- Codificador 8:3.
- 4 filtros paso bajo y su electrónica asociada
- 2 amplificadores operacionales LM324N (4 canales cada uno).

En la ilustración 40 se pueden ver las distintas etapas por las que va pasando la señal hasta llegar al microcontrolador.

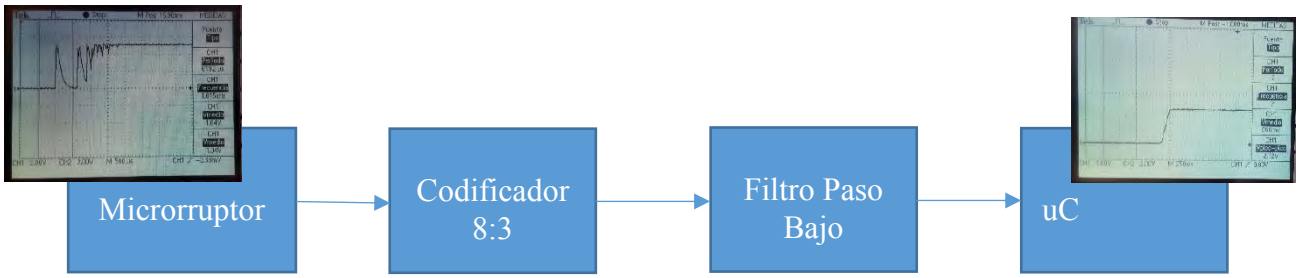


Ilustración 40. Etapas del procesamiento de señal del subsistema microrruptores [Diagrama].

En los anexos está el esquemático completo del subsistema microrruptores.

Alternativa de diseño

Una alternativa a los microrruptores que se planteó "a posteriori" fue la de añadir a los motores paso a paso un codificador rotatorio (encoder). De esta forma se conseguiría un control de los motores paso a paso en bucle cerrado y no existiría pérdida de pasos por lo que la precisión sería total. Sería un elemento de seguridad igualmente válido y más económico. Sin embargo, esta solución quedó descartada principalmente por tres motivos:

- La alternativa surgió cuando la fase de diseño del subsistema de los microrruptores estaba casi finalizada.
- Hubieran sido necesarias, en función de la resolución del codificador rotatorio que se hubiera escogido, hasta 4 líneas más por cada motor.
- La criticidad del sistema se produce al final del carril, y los microrruptores ya la resuelven.

En los anexos se puede acceder a los esquemáticos completos de los microrruptores y su electrónica asociada. Y en el apartado anterior su descripción a nivel global.

2.3.3. Sistema de audio.

Descripción

Para la implementación del sistema de audio se ha escogido el módulo **DFPlayer Mini** de **DFRobot** (ilustración 41). Se trata de un módulo MP3 que se controla a través de un puerto serie, vía USB, o a través de botonera. Cuenta con un DAC integrado y un amplificador de 3W para salida directa hacia altavoces de 3W/8ohm. Posee un zócalo para uSD desde el que se puede reproducir, mediante el envío de comandos por el puerto serie del microcontrolador, audios en formato .MP3 o .WAV.

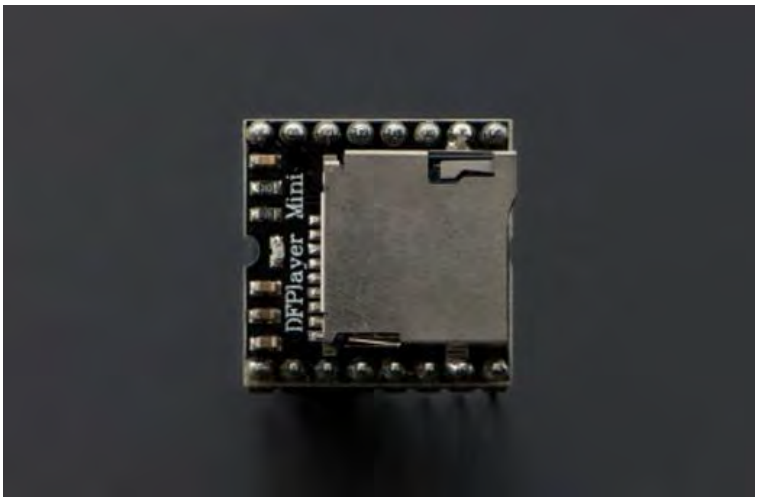


Ilustración 41. Módulo DFPlayer de DFRobot [Fotografía]. *(15)*

Los altavoces escogidos son de 4W y 4ohm y están distribuidos en serie, por lo que su impedancia total es de 8ohm. Se escogieron dos en lugar de uno para repartir mejor el sonido. Se trata del altavoz **ABS-230-RC**, de **Pro Signal** (ilustración 42).

Ilustración 42. Altavoz ABS-230-RD de Pro Signal [Fotografía]. *(16)*

El esquema final de este subsistema se muestra en la ilustración 43:

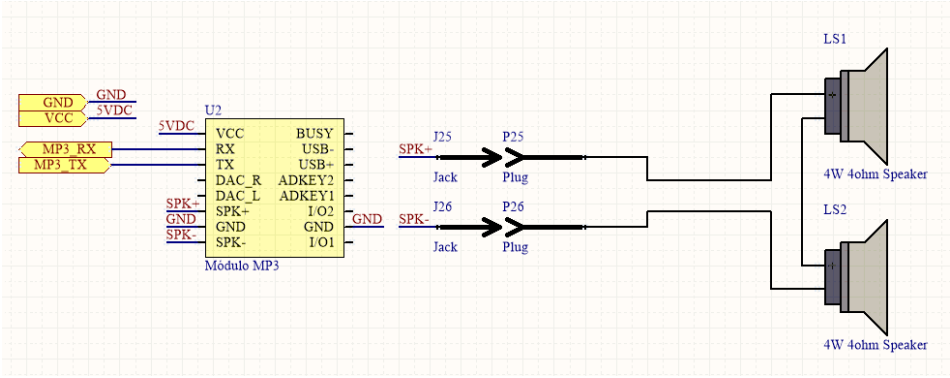


Ilustración 43. Esquema eléctrico del subsistema de audio [Esquema].

Alternativa de diseño

Se planteó en un primer momento como solución la generación del audio a través de un DAC, que sería controlado mediante el microcontrolador con una interfaz concreta en función del DAC escogido (por lo general suele ser i2c). A continuación, se debería añadir un amplificador para ya conectar directamente los altavoces. La fase de diseño en este punto se hubiera complicado bastante. Además, se hace necesaria también una forma de guardar los audios para transmitirlos. La nota de aplicación AN3126 de STMicroelectronics muestra una solución concreta para los microcontroladores STM32. En la ilustración siguiente se puede ver el esquema que tendría este sistema:

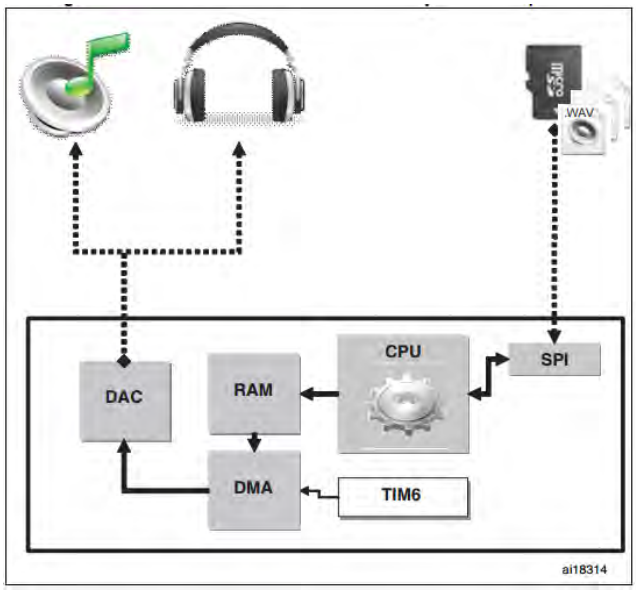


Ilustración 44. Solución para reproducir audio mediante microcontroladores STM32 [Esquema]. *(17)*

Exceptuando el hecho de que el esquema se aplica para un microcontrolador con DAC integrado, aplicando la misma idea a la solución planteada daría los mismos resultados.

ST ofrece en su página web un firmware demostración para implementar esta solución (18). Aun así, resulta evidente la complejidad que tiene llevar a cabo el desarrollo de esta parte del proyecto en comparación con la solución finalmente adoptada:

- Habilitar el Acceso Directo a Memoria (DMA) del microcontrolador para la transmisión y reproducción del audio en paralelo con la ejecución del programa.
- Implementar comunicación spi y/o i2c para la lectura de datos desde la uSD y la transmisión de estos al DAC.

En los anexos se puede acceder a los esquemáticos completos del sistema de audio. Y en el apartado 2.2 su descripción a nivel global.

2.3.4. Sistema de recepción por radiofrecuencia.

Descripción

La solución adoptada para la transmisión/recepción de datos desde el Bloque Rampa al Bloque Carriles es la compuesta por el conjunto codificador-transmisor-receptor-decodificador[5] ofrecida por **RFSolutions**:

- Codificador: **RF600E**
- Transmisor: **AM-RT4-433**
- Receptor: **AM-HRR30-433**
- Decodificador: **RF600D**

Se adopta la banda de frecuencia de 433MHz desde un principio porque no se requiere licencia para operar en ella. La transmisión utiliza un método de encriptado llamado *KeeLoq*, de **Microchip**. *KeeLoq* está basado en la codificación Manchester, cuya característica principal está en la combinación del flujo de datos y el reloj en una sola señal, de forma que se produce una autosincronización (19). En la transmisión basada en *KeeLoq*, junto con la información que se manda, se incluye además un chequeo por redundancia cíclica (CRC) garantizando que dicha información llega de forma correcta al receptor (20).

El **codificador** es un integrado con encapsulado PDIP-8. Tan solo requiere la conexión de cuatro entradas y el resto de la circuitería RF (transmisor más antena). La transmisión se activa automáticamente cuando detecta un cambio en una de sus cuatro entradas, tras un retardo de 6.5 ms.

El **transmisor** es capaz de mandar datos a una velocidad de hasta 4kHz, y puede transmitirlos directamente desde el microcontrolador por un puerto serie, o (como en la solución aplicada) utilizar un codificador. No precisa de electrónica auxiliar más allá de antena por lo que su implementación es muy sencilla.

[5] La descripción detallada del conjunto codificador-transmisor se puede encontrar en la memoria de Sergio Aguilera Sevilla.

El **receptor** (ilustración 45) es un módulo capaz de recibir e interpretar datos de cualquier transmisor AM que trabaje en su misma frecuencia. A través de un comparador interno transforma la señal, que previamente demodula, a niveles TTL para que esta pueda se pueda manipular digitalmente. Al igual que el receptor, toda la electrónica necesaria está embebida en el módulo, por lo que su implementación es muy sencilla.

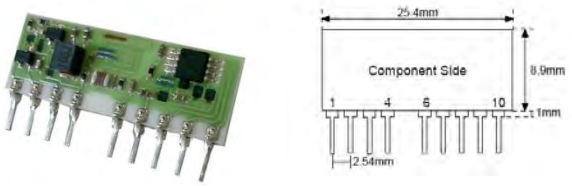


Ilustración 45. Módulo Receptor AM-HRR30-433 [Fotografía].

Siguiendo las indicaciones de la hoja de características:

- Los pines 1, 7 y 10 corresponden a Vcc (5V).
- Los pines 2, 4 y 6 van directos a GND.
- El pin 3 (DATA IN) se conecta a la antena. Asemejándola a una antena tipo Marconi (21), su longitud viene determinada por la longitud de onda de la frecuencia a la que se trabaja:

$$\lambda = \frac{c}{f} = \frac{300 \cdot 10^6\ [m/s]}{433MHz} = 0{,}693m$$

  De donde se obtiene que la longitud *l* de la antena es de:

$$l = \frac{\lambda}{4} = 17{,}3\ cm$$

- El pin 9 conecta directamente con el decodificador RF600D

El **decodificador** (Ilustración 46) es un dispositivo con encapsulado DIP18 cuya finalidad es recibir una trama de datos en serie a niveles TTL con encriptación *KeeLoq* (no se entiende el uso de este integrado sin el RF600E, que es el que se encarga de aplicar esta encriptación a la trama) y decodificarla para mandarla al microcontrolador (ya sea en serie o en paralelo). Además, como está pensado para usarse en transmisiones inalámbricas, es capaz de reconocer hasta 7 codificadores RF600E guardando en memoria interna sus números de serie. Añadiendo una EEPROM al sistema puede llegar a guardar hasta 48 codificadores.

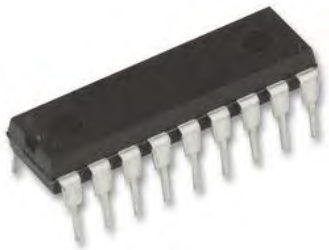


Ilustración 46. Decodificador RF600D [Fotografía]. *(22)*

Para el correcto funcionamiento del dispositivo, es necesario añadir la siguiente electrónica auxiliar al diseño:

- Un interruptor, que se usa en el proceso de grabación del número de serie del codificador.
- Un led, que indica cuándo está recibiendo información y el estado de memorización durante el proceso de grabado.
- Una resistencia cuya posición determina qué tipo de transmisión se está realizando, si AM o FM.
- Una resistencia cuya posición determina qué tipo de tratamiento se le va a dar a la salida paralelo del decodificador.

El decodificador tiene 4 salidas (OP4, OP3, OP2 y OP1) que corresponden a cada una de las 4 entradas del codificador (S3, S2, S1 y S0). Un cambio en cualquiera de las cuatro entradas del codificador se ve reflejado en su correspondiente salida del decodificador tras un tiempo de unos 160 ms. En función de dónde se coloque la resistencia en el pin LKIN se tienen dos modos de funcionamiento:

- *Latching:* El estado en la salida cambia con cada transmisión de señal válida.
- *Momentary:* La salida se mantiene durante toda la duración de la transmisión de señal válida, indiferentemente de que durante esta cambie el valor de nuevo.

La transmisión también se produce en serie, a través del pin SD1. Las características de esta transmisión serie son:

- 9600 baudios por segundo.
- Tamaño del dato de 8 bits.
- Un bit de stop.
- Sin paridad.

Durante la transmisión, son enviados 10 bytes de información (ilustración 47):

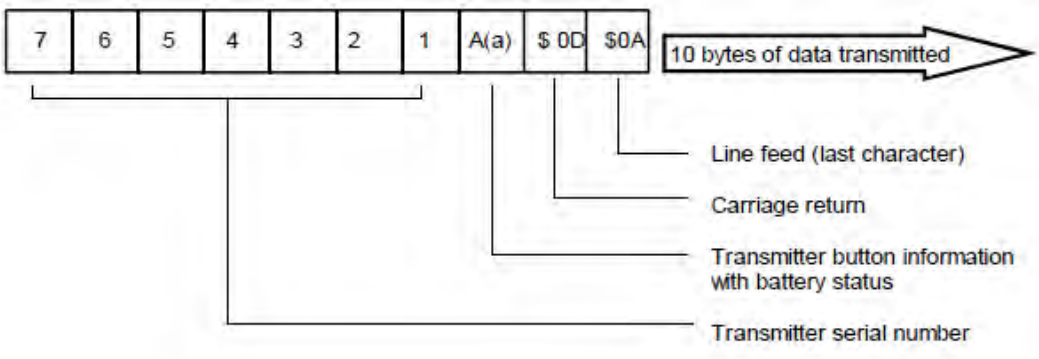


Ilustración 47. Trama de datos que se transmite desde el RF600E al RF600D. [Diagrama].

Los 7 primeros bytes corresponden al número de serie del codificador que está transmitiendo la información. El byte número 8 es el que transmite la información del estado de las entradas del codificador. Los dos últimos bits corresponden al retorno de carro y salto de línea.

El byte 8 es un carácter cuyo valor en código ASCII es 'A' cuando el estado de todas las entradas del codificador es 0. Cuando todas están a uno el valor es 'P'. Siguiendo un orden binario, se tiene la tabla de la ilustración 48:

SD1	S3	S2	S1	S0
A	0	0	0	0
B	0	0	0	1
C	0	0	1	0
D	0	0	1	1
E	0	1	0	0
F	0	1	0	1
G	0	1	1	0
H	0	1	1	1
I	1	0	0	0
J	1	0	0	1
K	1	0	1	0
L	1	0	1	1
M	1	1	0	0
N	1	1	0	1
O	1	1	1	0
P	1	1	1	1

Ilustración 48. Tabla de código ASCII de la salida serie del decodificador RF600D [Tabla].

La información será leída en serie dado que luego es más sencillo interpretarla desde el microcontrolador. El objetivo es asignar en el Bloque Rampa un comando distinto a cada letra, y en función del comando actuar en consecuencia en el Bloque Carriles (se detallará mejor en el apartado 3.2.1).

Teniendo en cuenta las necesidades del diseño y la hoja de características del decodificador, tenemos que:

- Los pines de alimentación 4(Vcc) y 5(Vss) van a 5V y Gnd, respectivamente.
- LB, EDAT, ECS, LKIN, OP4, OP2 y OP1 no se utilizan, por lo que se dejan sin conectar.
- SLEEP se pone a nivel alto para un modo de funcionamiento normal.
- IN se conecta directamente al receptor.

- LRN va conectado a un interruptor y al led a través de una resistencia de 1kohm.
- ECLK va conectado a tierra a través de una resistencia de 22kohm para establecer modo de trabajo AM.
- SD1 se conecta directamente a una entrada serie del microcontrolador.

Por tanto, el esquema de este subsistema es el que se muestra en la ilustración 49:

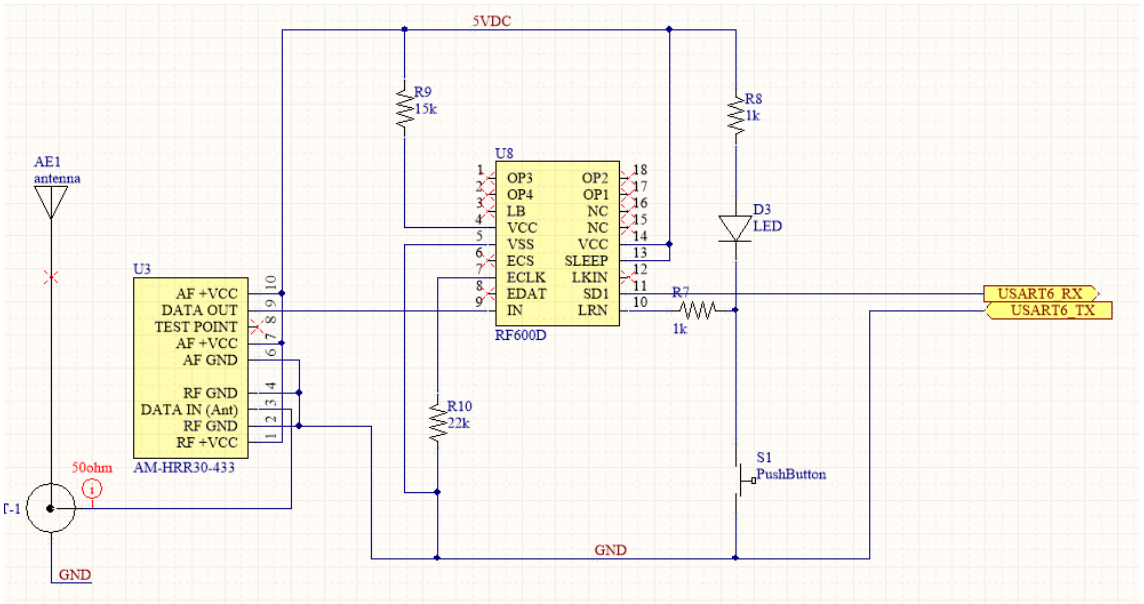


Ilustración 49. Esquema eléctrico del subsistema de recepción por radiofrecuencia [Esquema].

Alternativa de diseño

Debido a la eficacia que la solución finalmente adoptada tuvo en proyectos anteriores, no se estudió la implementación de otros sistemas. En el proyecto consultado (23) también surge la necesidad de implementar una comunicación unidireccional y en él se comprueba, en las pruebas de transmisión que realizaron a través de pulsación de botones, que funcionó correctamente. De una forma parecida, en el presente proyecto se implementan unas pruebas mediante pulsación de botones (apartado 3.1.4).

En los anexos se puede acceder a los esquemáticos completos del sistema de recepción por radiofrecuencia. Y en el apartado anterior su descripción a nivel global.

2.3.5. Microcontrolador.

Descripción

El Bloque Carriles está gobernado por una placa de desarrollo de STMicroelectronics: la NUCLEO-F411RE (ilustración 50) que entre otras características tiene:

- Microcontrolador embebido STM32F411RE, de 32 bits y 64 pines.
- Posibilidad de depurar programa a través del ST-LINK, con conexión usb tipo mini-b.
- Dos pulsadores ("reset" y "user").
- Posibilidad de alimentar a 3,3V, a 5V o a 12V.

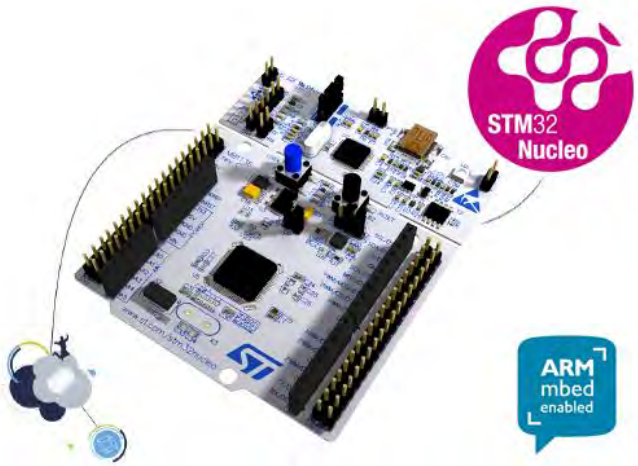


Ilustración 50. Placa de desarrollo NUCLEO-F411RE. [Fotografía].

El microcontrolador que tiene embebido posee una memoria flash de 512Kbytes, lo cual es más que de sobra para poder programar sin preocuparse del espacio que pueda ocupar. Capaz de trabajar a una frecuencia de hasta 26 MHz, tiene un consumo en funcionamiento normal de 100uA y de 2.4uA en modo Standby. Posee 11 timers, hasta 81 GPIO's[6] (de los cuales la mayoría toleran hasta 5V), interfaz de comunicación i2c, spi, usart…

Muchas de estas características hacen que sea un microcontrolador ideal para trabajar con él. Así todo, uno de los motivos principales por el que se escogió como solución definitiva fue su compatibilidad con el software STM32CubeMX. Este programa permite, previa selección del microcontrolador que se desee (debe ser un STM de 32 bits), la modificación de parámetros tales como configuración de los pines (como I/O -pullup, pulldown-, timers, USART...), reloj que gobierna el micro y cada uno de los buses de cada periférico.

Una vez modificado los parámetros, el programa genera el código de inicialización en C necesario cargar en el microcontrolador para configurar todos los periféricos según las

[6] *GPIO: General Purpose Input/Output*

necesidades del proyecto, lo cual permite la liberación de una carga importante de trabajo por parte del programador.

En la ilustración 51 se puede observar cómo es la interfaz del programa. Pin a pin se puede seleccionar qué función específica se desea que tenga. En el cuadro lateral se puede activar cada uno de los periféricos que se desee (siempre que los pines no hayan sido ya utilizados para otro propósito -en tal caso se muestra una X al lado del nombre del periférico-).

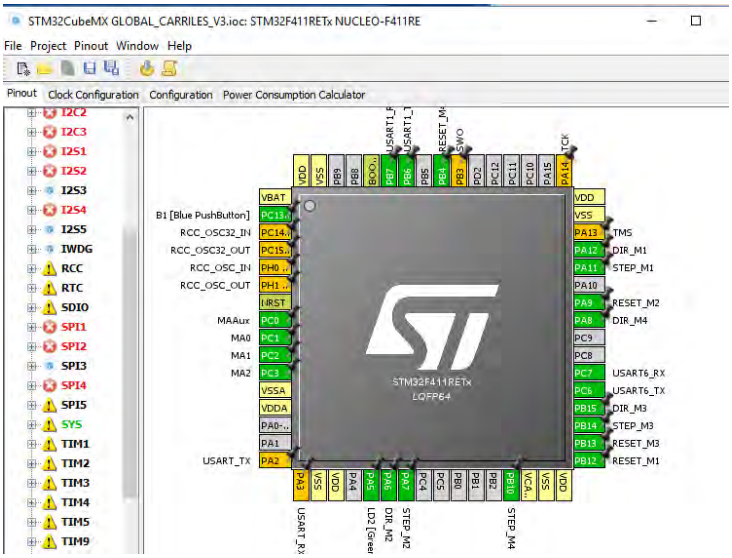


Ilustración 51. Interfaz de configuración de programa STM32CubeMX [Imagen]

Configuración

En lo referente al **reloj** que va a gobernar el micro, dejamos la configuración que viene por defecto. Se utiliza el oscilador HSI[7] (16MHz) para, a través de un dispositivo de realimentación de frecuencia y fase llamado PLL[8], establecer la frecuencia del reloj del sistema (SYSCLK) en 84MHz. De esta frecuencia dependen todas las frecuencias de trabajo del resto de periféricos, tal y como se muestra en la ilustración 52.

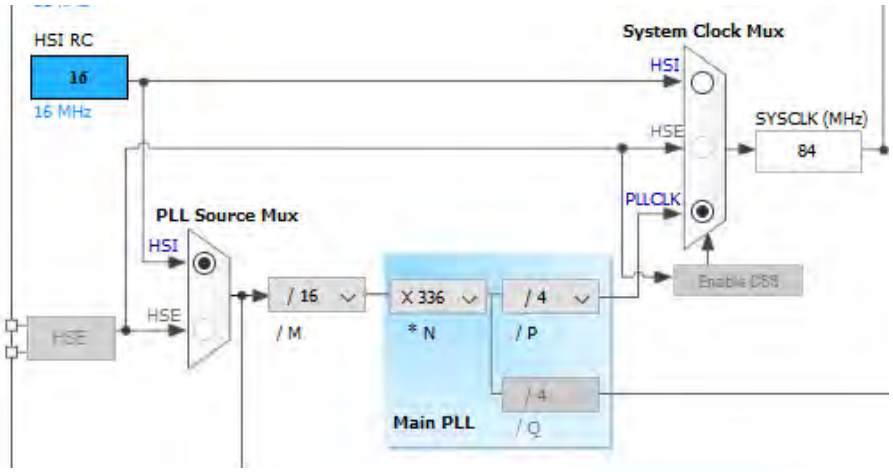


Ilustración 52. Configuración del reloj del sistema. [Imagen]

[7] HSI: High Speed Internal.
[8] PLL: Phase-Locked Loop

A continuación, y teniendo en cuenta las necesidades expuestas anteriormente para cada uno de los subsistemas anteriores, se detalla la configuración de cada pin y uso de los periféricos (ilustración 53).

Subsistema	Etiqueta	Pin	Modo	Pull-up/Pull-Down
Motores PAP	STEP_M1	PA11	**GPIO** Ouput Push Pull	Pull-Down
	DIR_M1	PA12	**GPIO** Ouput Push Pull	Pull-Down
	RESET_M1	PB12	**GPIO** Ouput Push Pull	Pull-Down
	STEP_M2	PA7	**GPIO** Ouput Push Pull	Pull-Down
	DIR_M2	PA6	**GPIO** Ouput Push Pull	Pull-Down
	RESET_M2	PA9	**GPIO** Ouput Push Pull	Pull-Down
	STEP_M3	PB14	**GPIO** Ouput Push Pull	Pull-Down
	DIR_M3	PB15	**GPIO** Ouput Push Pull	Pull-Down
	RESET_M3	PB13	**GPIO** Ouput Push Pull	Pull-Down
	STEP_M4	PB10	**GPIO** Ouput Push Pull	Pull-Down
	DIR_M4	PA8	**GPIO** Ouput Push Pull	Pull-Down
	RESET_M4	PB4	**GPIO** Ouput Push Pull	Pull-Down
Microrruptores	MA2	PC3	**GPIO** EXTI Flanco Subida	n/a
	MA1	PC2	**GPIO** EXTI Flanco Subida	n/a
	MA0	PC1	**GPIO** EXTI Flanco Subida	n/a
	MAAux	PC0	**GPIO** EXTI Flanco Subida	n/a
Audio	USART1_RX	PB6	**GPIO** Alternate Function	Pull-Up
	USART1_TX	PB7	**GPIO** Alternate Function	Pull-Up
Radiofrecuencia	USART6_TX	PC6	**GPIO** Alternate Function	Pull-Up
	USART6_RX	PC7	**GPIO** Alternate Function	Pull-Up

*Ilustración 53.*Tabla con todos los pines del Bloque Carriles y su configuración [Tabla].

Por último, la configuración que se ha aplicado a ambos periféricos USART1 y USART 6:

- Baudios: 9600 por segundo
- Longitud de la palabra: 8 bits (incluyendo bit de paridad).
- Paridad: no
- Bits de stop: 1.

En los anexos se puede acceder a los esquemáticos completos del microcontrolador radiofrecuencia. Y en el apartado anterior su descripción a nivel global.

2.3.6. Alimentación.

En lo referente a la alimentación, el sistema necesita:

- 5V para el microcontrolador y la mayor parte de la lógica.
- 12V para los motores paso a paso.
- 3,3V para el amplificador operacional.

Los 3,3V se pueden sacar de la placa de desarrollo escogida, concretamente del pin +3V3, que puede entregar hasta 500mA (corriente limitada por el regulador de tensión que tiene la placa de desarrollo).

Para los 5V y los 12V hay que adquirir una fuente de alimentación, cuya potencia vendrá determinada por los motores, ya que son los que más corriente van a exigir. Poniendo que el peor caso que se va a dar es aquel en el que los cuatro motores se están moviendo a la vez, se necesitaría una fuente de al menos:

$$P = 4 \cdot (12V \cdot 0{,}5A) \cdot k = 48W$$

Sea k=2 un factor de seguridad.

Por otro lado, necesitamos que la fuente tenga dos salidas, una de 5V y otra de 12V. La solución escogida es la que ofrece **Traco Power** y su fuente de alimentación **TXL 060-0512DI** (ilustración 54). Puede entregar hasta 60W y tiene dos salidas, una de 5V-8A y otra de 12V-4A.

Ilustración 54. Fuente de alimentación Traco Power [Fotografía]. *(16)*

Para encender y apagar la fuente desde fuera del chasis, se precisa de un botón de montaje en panel. Se selecciona el botón **MC34231-091-72** de **Multicomp**, por ser retroiluminado, por soportar hasta 250V de alterna y por ser compatible con los conectores hembra aéreo de la serie FASTON 250 de TE Connectivity. Su conexionado es el que se muestra en la ilustración 55:

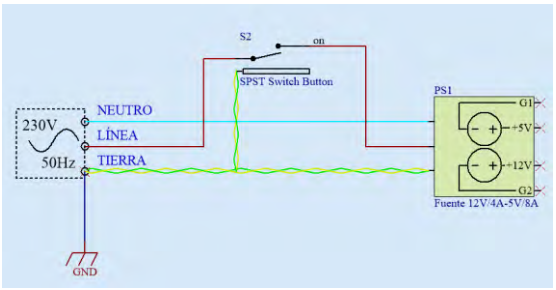


Ilustración 55. Esquema eléctrico del conexionado del botón MC34231-091-72 [Fotografía].

En los anexos se puede acceder a los esquemáticos completos del microcontrolador radiofrecuencia. Y en el apartado anterior su descripción a nivel global.

2.4. Diseño de las placas de circuito impreso.

2.4.1. Conceptos y elementos básicos comunes a ambas placas.

Capacidades Técnicas

Para el diseño de las PCB's se contactó con *2CISA*[9], por ello el diseño ha estado condicionado por sus capacidades técnicas y por el estándar con el que trabajan, que por otro lado es el que más se usa en el ámbito profesional. A continuación, se describen brevemente los requisitos técnicos que se debieron cumplir en el diseño de las PCB's:

- Un grosor total máximo de 1.6mm +/- 10% para una placa de 4 capas.
- Grosor máximo de las pistas de cobre y de los planos de 35um.
- Material del dieléctrico de tipo FR-4 y grosor de 1,2mm. El FR-4 consiste en un aislante eléctrico formado por capas de fibra de vidrio pegadas entre sí por resina Epoxi (24).
- Capas intermedias entre planos y pistas del mismo material que el núcleo, pero con un grosor de 0,173 mm (25).

En la figura 56 se recogen estos puntos:

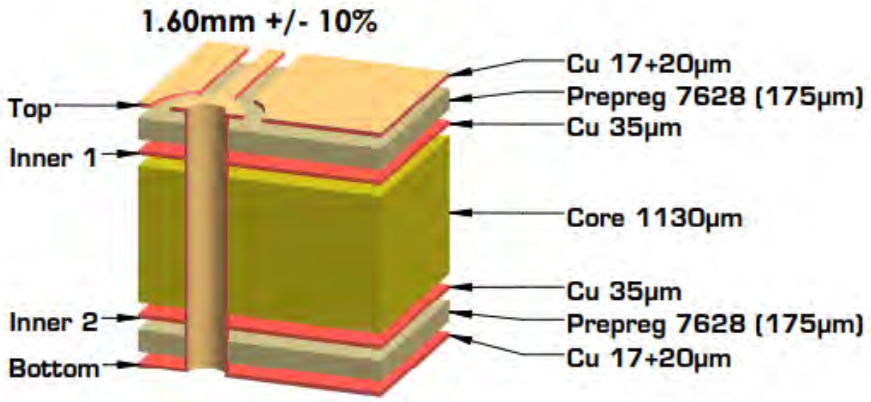


Ilustración 56. Capas y sus grosores y materiales para PCB de 4 capas y 1.6mm de grosor [Imagen]. *(26)*

- Tamaño mínimo del diámetro del agujero en taladros metalizados de 0,3mm.
- *Aspect Ratio (AR)* menor o igual que 6:

$$AR = \frac{Grueso\ PCB}{Taladro\ mínimo} = \frac{1{,}6mm}{0{,}3mm} = 5{,}333$$

- Tamaño mínimo de la corona en taladros metalizados/vías de 0,2mm.
- Anchura mínima de las pistas de 0.3mm.

[9] Empresa afincada en Barcelona de fabricación de circuitos impresos. Se encargaron de la fabricación y envío de ambas placas.

Componentes utilizados

Hay elementos que, aunque no fuera necesario determinar durante la fase de prototipado, sí que han sido necesarios a la hora de diseñar las PCB's, a continuación se enumeran y describen estos elementos y la razón de su elección.

En lo referente a elementos pasivos tales como resistencias y condensadores, se van a utilizar componentes de montaje superficial, con el objetivo de reducir en la medida de lo posible el tamaño de la PCB, ya que influye directamente en el precio de esta. Dentro de los distintos tamaños de resistencias/condensadores se van a utilizar aquellos con encapsulado 1206, son de los más grandes y por tanto fácilmente manipulables (ilustración 57).

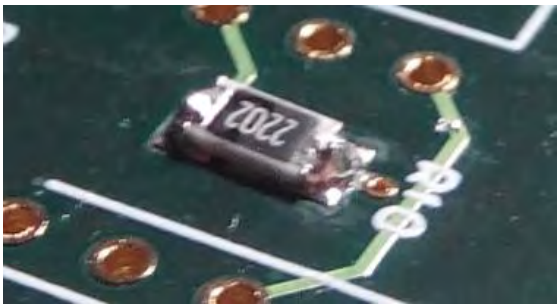


Ilustración 57. Detalle de una resistencia de montaje superficial 1206 [Fotografía].

El valor de las resistencias y condensadores utilizados están descritos en los apartados anteriores. Sin embargo, se ha tenido también en cuenta la inclusión de **condensadores de desacoplo**. Los condensadores de desacoplo se colocan lo más próximo posible al integrado y su finalidad es la de filtrar el ruido indeseado que se acopla en los niveles de alimentación. Este ruido normalmente puede provenir de diversas fuentes: de la propia fuente de alimentación, de fuentes externas como consecuencia de un incorrecto aislamiento de la placa, de otros integrados, etc.

Para conectar todos los cables que van desde los distintos elementos de los subsistemas de ambos bloques (altavoces, microrruptores, motores paso a paso, servomotores, fuente de alimentación…) se utilizan los siguientes conectores:

- Conector macho para PCB **63824-1** y su homólogo hembra aéreo **60838-1**, ambos de la serie FASTON 250 de **TE Connectivity**. (ilustración 58).

Ilustración 58. Conectores macho y hembra de la serie FASTON 250 de TE Connectivity [Fotografía].

- Para conectar las placas de desarrollo se han utilizado conectores de 19x2 posiciones. Concretamente de **SAMTEC**, el CES-119-02-T-D (ilustración 59).

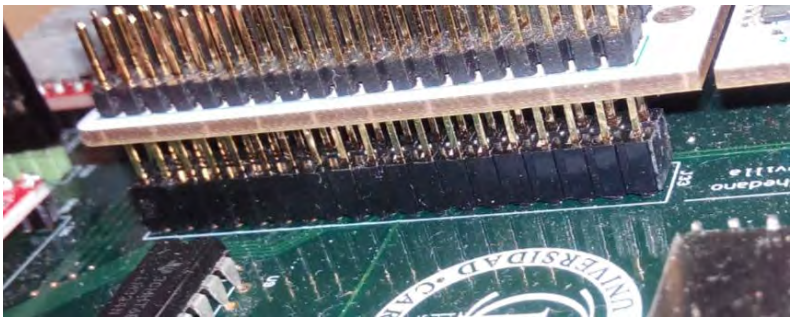


Ilustración 59. Conector CES-119-02-T-D de SAMTEC [Fotografía].

- En lo referente a los motores paso a paso, como estos venían con conectores hembra aéreos con paso estándar de 2.54mm, se han buscado sus homólogos macho para PCB (**826926-4** de **TE Connectivity**).

- Para los servomotores, se han utilizado conectores por desplazamiento del aislante (IDC) hembra de 3 posiciones para crimpar los cables y otro conector macho para conexión directa con la PCB (el **3-640442-3** y el **640454-3**, de la serie MTA-100, ambos de **TE Connectivity,** ilustración 60).

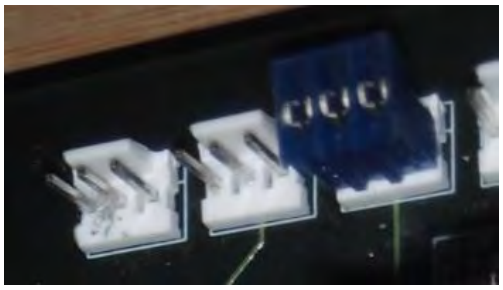


Ilustración 60. Conectores para los servomotores [Fotografía].

- Para unir el LCD del Bloque Rampa a la placa se han utilizado los conectores macho **2213S-16G** y hembra **2214S-16SG-85** de **Multicomp.**

- El conector seleccionado para la inserción de los cables de alimentación ha sido el **MPT 0,5/ 2-2,54** de **Phoenix Contact** (ilustración 61). Es un conector de dos contactos que soporta hasta 6A de corriente, lo cual es suficiente ya que ninguna de las placas va a necesitar más corriente en ningún momento.

Ilustración 61. Conectores MPT 0.572-2,54 de Phoenix Contact [Fotografía].

Además de los conectores, resistencias y condensadores, hay otros elementos necesarios para implementar el sistema completo:

- Para los módulos de radiofrecuencia de ambas antenas se hace necesario un conector especial de RF con el objetivo de mejorar lo máximo posible la calidad de la señal. Por ello se elige un conector tipo UFL con una impedancia de 50 ohm, concretamente el **U.FL-R-SMT-1(10)** de **HIROSE** (ilustración 62), y una antena compatible con este adaptada para frecuencias de 433MHz (**66089-0430** de **ANAREN**).

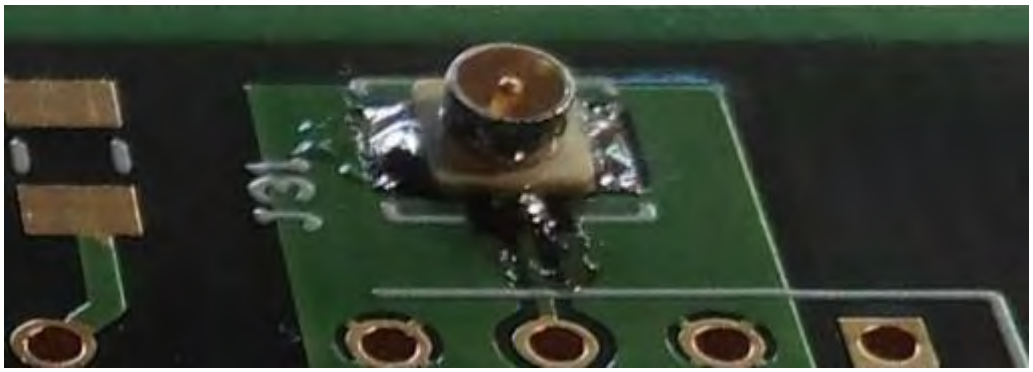
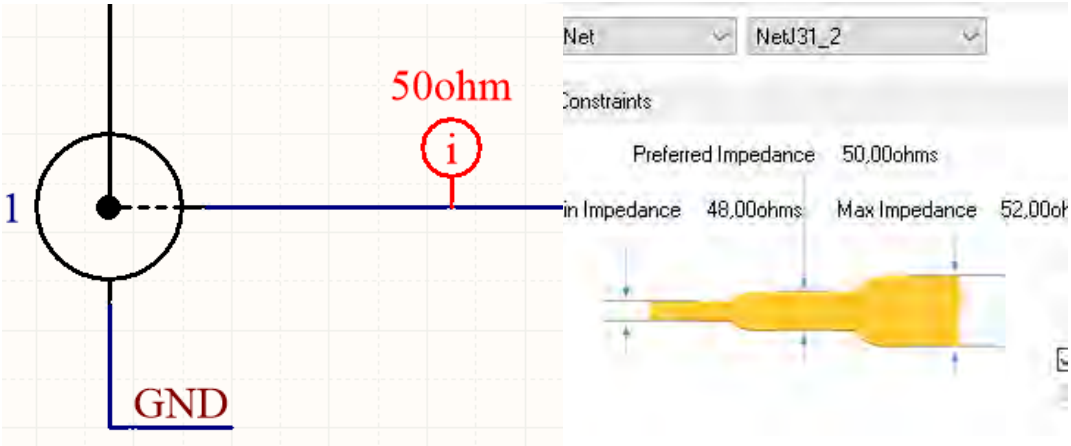


Ilustración 62. Conector UFL para la antena de los módulos de radiofrecuencia [Fotografía].

Durante toda la línea de transmisión, es necesaria una **adaptación de impedancias**. Es decir, la impedancia de entrada, la de la línea de transmisión y la de carga deben ser iguales para evitar reflexiones de señal y con ello pérdida de información (27). Es necesario que el ancho de pista sea tal que su impedancia sea de 50 ohm. Esta anchura se puede determinar automáticamente desde el software utilizado para el diseño de las PCB's, aplicando una directriz de diseño (ilustraciones 63 y 64).



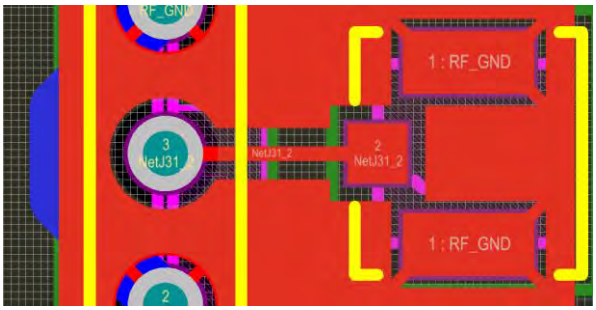


Ilustración 63. El ancho de la pista se ajusta al valor asignado mediante la directriz de diseño [Figura].

- Para el módulo receptor de radiofrecuencia se necesita un interruptor para el proceso de aprendizaje de un RF600E por parte del RF600D, tal y como se describió en el apartado 2.3.4. Se elige el interruptor de montaje en superficie **2468741** de **TE Connectivity**.

- Son necesarios un led para cada módulo de radiofrecuencia. Para este propósito sirve el **LYA67K-J2M1-26** de **OSRAM** (ilustración 65)**.** De 1.8V y 2mA.

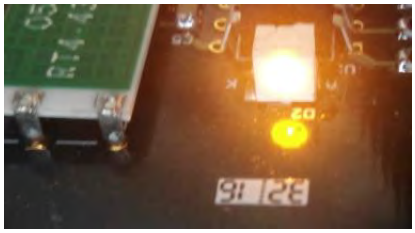


Ilustración 64. Led encendido durante proceso de aprendizaje de RF600D [Fotografía].

Para poder observar visualmente si la fuente está conectada, se va a conectar este mismo led justo después de los conectores de la alimentación. Para limitar la corriente que circula a través de él, debe llevar una resistencia asociada en serie, cuyo valor viene determinado por:

$$R = \frac{Vcc - Vled}{Iled}$$

En el caso de la fuente de 12V este valor es de 5kohm. Se aprovechan dos de 15kohm para tener una resistencia equivalente de 7.5kohm, que también servirán. El valor de la resistencia para la fuente de 5V es de 1.5kohm, como no se usan resistencias de este valor en todo el proyecto, se aprovechan las de 1kohm. En la ilustración 66 se muestra un esquema.

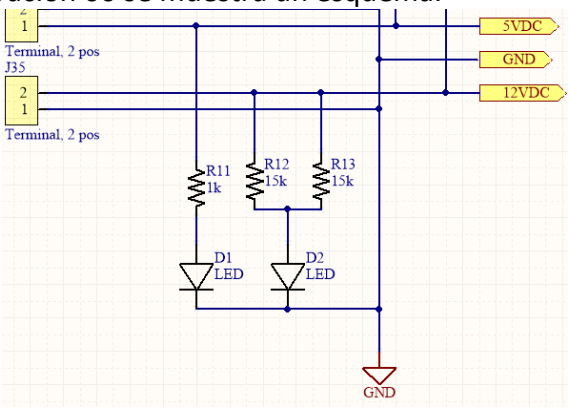


Ilustración 65. Esquema de los leds de alimentación y resistencias asociadas [Esquema].

En lo referente a la fijación mecánica de las placas sobre los chasis. Para fijar la placa del Bloque Carriles al chasis se utilizan cuatro separadores con base adhesiva y se coloca uno en cada esquina (**06.21.809** de **ETTINGER**). Se colocan en cada esquina cuatro agujeros de montaje de 3.2mm para la inserción y fijación de los separadores en la PCB.

Para la fijación de la placa del Bloque Rampa en el chasis de la rampa se utilizan cuatro separadores con terminación atornillable con rosca M4 de 25mm (**06.84.256** de **ETTINGER**), lógicamente se fijan con un tornillo de métrica 4. Necesitan cuatro agujeros en la PCB de 4mm para su inserción, uno en cada esquina.

En esta placa también era necesario fijar el LCD de forma que sobresaliera lo suficiente como para que estuviera en una posición correcta sobre el panel de control, es por ello que se adoptó la solución de colocar un separador hexagonal metálico de 4mm con una rosca interior del mismo diámetro que los agujeros de montaje del LCD escogido, por tanto se adopta la solución **R25-1001402** de **HARWIN.** Se necesitan además 8 tornillos de métrica 2.5 y 8 arandelas para lograr finalmente el montaje que se muestra en la ilustración 67.

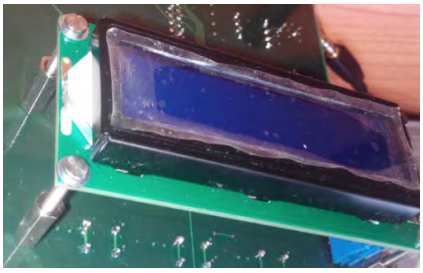


Ilustración 66. Montaje del LCD sobre la placa del Bloque Rampa [Fotografía].

Por último, como durante la fase de diseño de las PCB's hubo varios aspectos que no terminaron de determinarse a tiempo, se fabricó especialmente para poder ser modificada en función de las necesidades finales. Un ejemplo de ello es el uso de puentes de soldado (*Solder Bridges* -SB-) en la interfaz del botón de accionamiento de la rampa (en la placa del Bloque Rampa). En la ilustración 67 se observa que la decisión final fue la de soldar el SB10 para hacer que la pulsación provoque un flanco de subida.

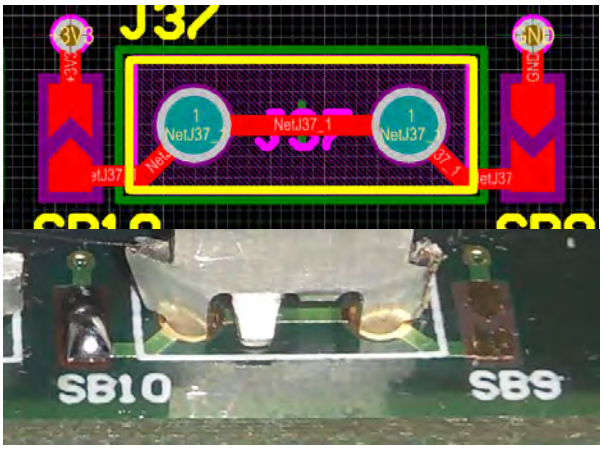


Ilustración 67. Fotografía y esquema de los terminales SB10 y SB9 [Imagen].

2.4.2. Proceso de diseño de las placas.

Para el diseño de las placas se ha precisado de la utilización del software *Altium Designer*. El proceso de diseño ha sido igual para ambas placas y ha consistido en lo siguiente:

- Diseño de los esquemas eléctricos de cada subsistema y elección de los componentes para las PCB's.
- Creación de huellas de cada componente y exportación de los esquemas y huellas asociadas sobre sobre la PCB.
- Colocación de componentes y rutado.
- Generación de gerbers y documentación técnica y envío de estos a 2CISA.

En las siguientes ilustraciones se muestra el esquema de rutado y el resultado final para la placa del Bloque Carriles.

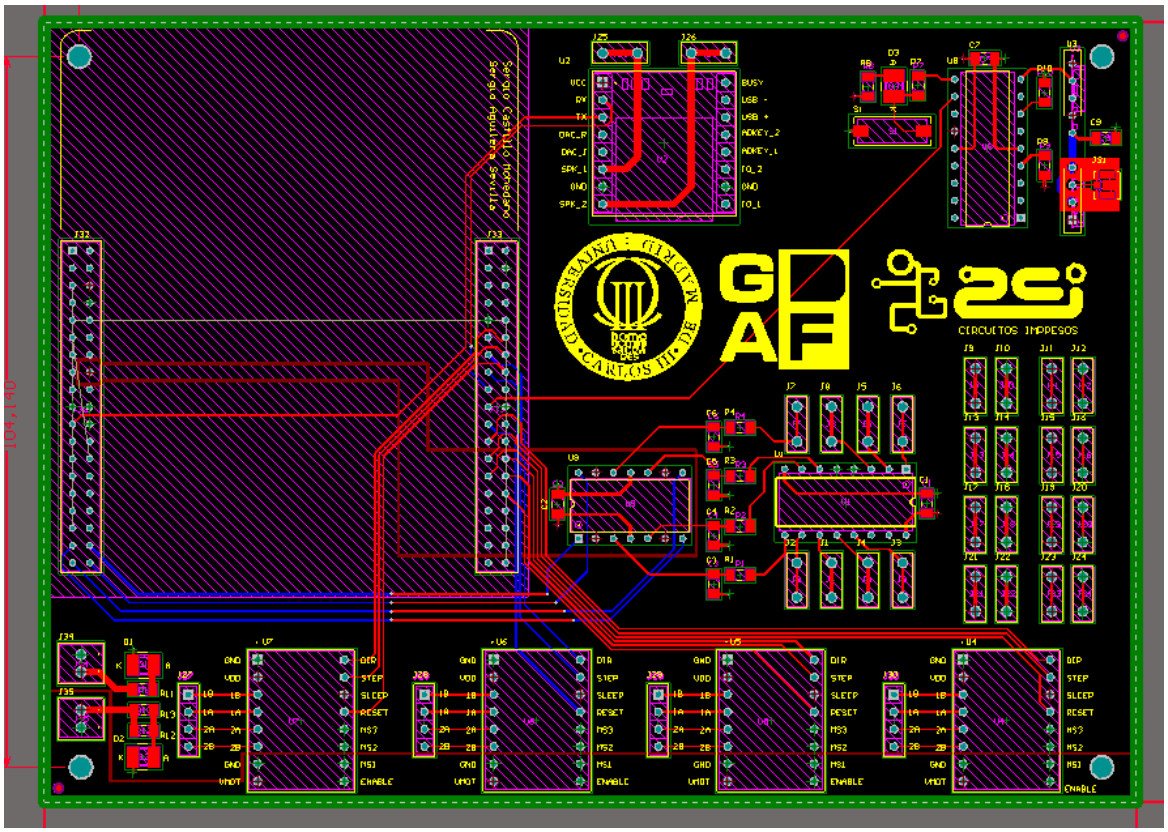


Ilustración 68. Esquema de rutado de la placa del Bloque Carriles [Esquema].

Ilustración 69. Vista inferior de la placa del Bloque Carriles [Fotografía].

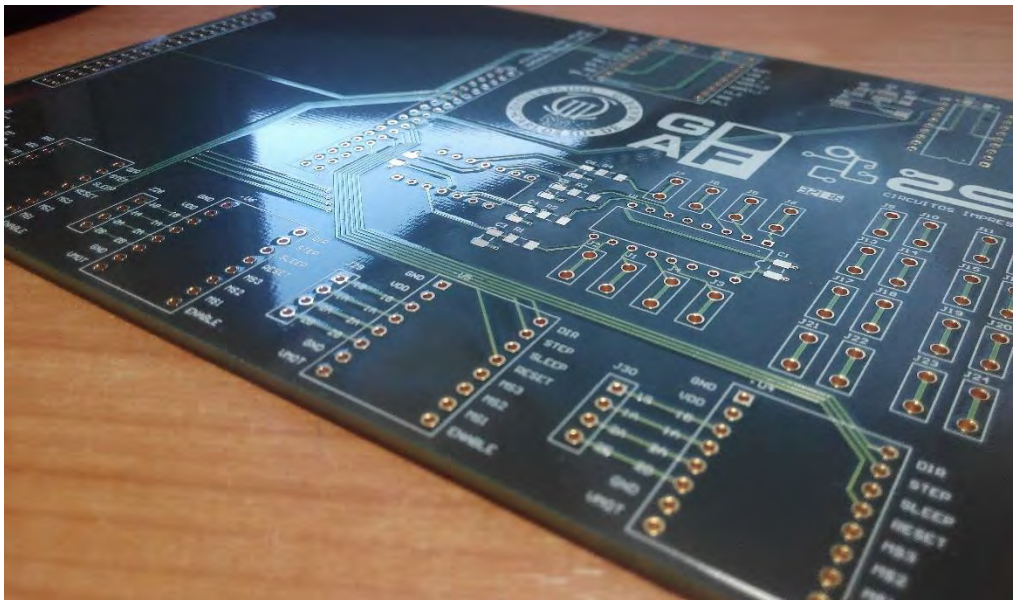


Ilustración 70. Vista superior de la placa del Bloque Carriles [Fotografía].

En los anexos se puede consultar la siguiente documentación referente a las placas de circuito impreso:

- Plano mecánico de la PCB del Bloque Carriles
- Plano de Situación de Componentes del Bloque Carriles.
- Plano Mecánico de la PCB del Bloque Rampa.
- Plano de Situación de Componentes del Bloque Rampa.

3. Capítulo 3. Implementación del sistema.

3.1. Pruebas del Bloque Carriles.

Después de determinar los elementos de cada subsistema se procedió a la realización de una serie de pruebas para asegurar el correcto funcionamiento de los mismos. A continuación se describen brevemente estas pruebas.

3.1.1. Motores paso a paso.

La verificación de control de los motores paso a paso con el módulo StepStick de RepRap consistió en la implementación de este en una placa de prototipado (ilustración 72) y la realización de un pequeño programa.

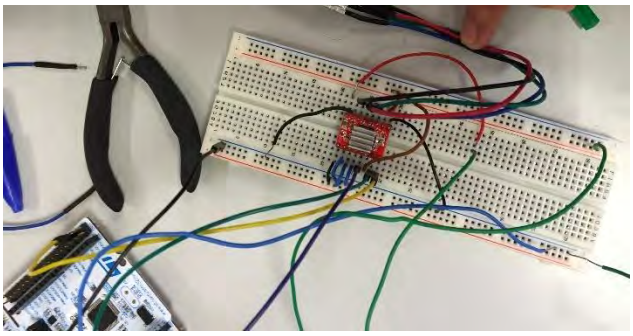


Ilustración 71. Implementación de StepStick de RepRap sobre placa de prototipado [Fotografía].

El programa escrito es muy sencillo y lo único que hace es mover el motor 360 grados aplicando un pulso cuadrado de 500Hz durante 400 ms. De forma que se aplican 200 pasos (200·1.8 grados = 360 grados). Una vez asimilado este concepto se diseñaron funciones más complejas para mover los motores una cantidad de grados proporcional a la distancia que se deseaba desplazar el muñeco.

```
001     while (1)
002     {
003           for (i=0;i<1;i++)
004           {
005                 HAL_Delay(1);
006                 HAL_GPIO_TogglePin(_STEP_GPIO_Port, _STEP_Pin);
007                 reset++;
008                 if (reset == 400)
009                 {
010                       reset = 0;
011                       HAL_GPIO_TogglePin(_RST_GPIO_Port, _RST_Pin);
012                 }
013           }
014     }
015   }
```

3.1.2. Microrruptores.

Para llevar a cabo el diseño de la electrónica de los microrruptores descrita en el apartado 2.3.2. se implementó un pequeño programa.

El programa consiste en, mediante la pulsación de los microrruptores, la activación o desactivación de una fila de leds como los de la página 73.

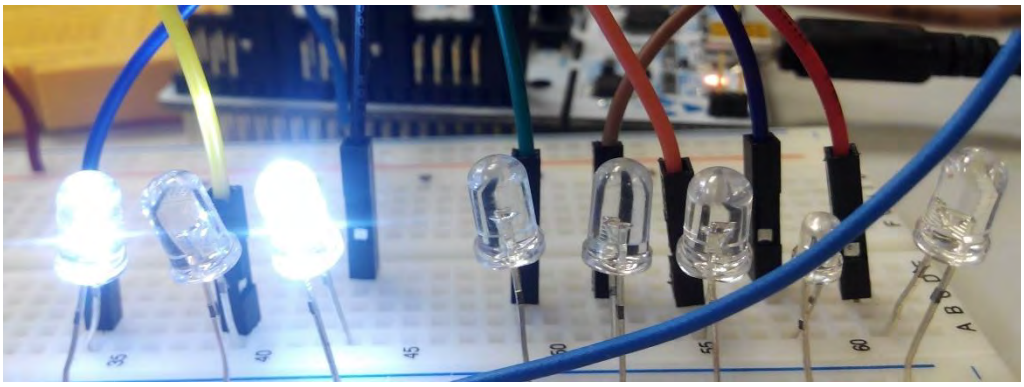


Ilustración 72. Fila de leds que se activan mediante pulsación de microrruptor [Fotografía].

Mediante tres variables globales declaradas como *extern* cuyo valor se altera dentro de las interrupciones en el fichero *stm32f4xx_it.c* se enciende un led u otro dependiendo de los valores de estas variables.

3.1.3. Sistema de Audio.

La implementación a nivel *hardware* fue tan sencilla como seguir las indicaciones del datasheet, el conexionado está ya descrito en el apartado 2.3.3. Sin embargo, fue necesario primero aprender a controlar el puerto USART del microcontrolador y segundo a escribir las librerías de comandos del módulo RFPlayer[10].

A través de un convertidor TTL-USB como el de la ilustración 74 se pudieron hacer pruebas y comprobar el correcto envío de las tramas no solo del microcontrolador al DFPlayer si no también de las tramas del módulo de radiofrecuencia.

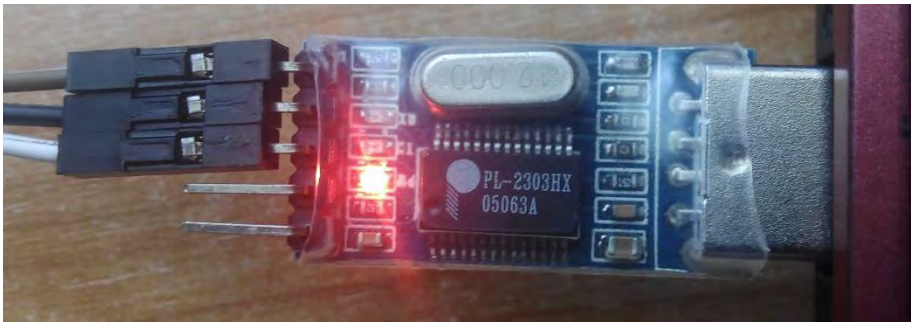


Ilustración 73. Convertidor TTL-USB

[10] Se encontraron unas librerías específicas (34) para este módulo que, previa modificación de las mismas para adecuación al microcontrolador, terminaron por servir para la aplicación.

En las ilustraciones 75 y 76 se describen las características que debe cumplir el puerto USART y el formato que tiene el comando que se envía al DFPlayer:

```
ASYNCHRONOUS SERIAL COMMUNICATION FEATURES
        for FPlayer Mini Device
-------------------------------------------

        9600 bauds per second
        Data Bits: 8
        Parity: none
        Stop Bits: 1
        Checkout: none
        Flow Control: none
```

Ilustración 74. Requisitos de la transmisión USART

```
FORMAT: 10Bytes Word Data Lenght
$S VER Len CMD Feedback para1 para2 checksum $O
```

$S	Start byte 0x7E	Each command feedback begins with $, which is 0x7E
VER	Version	Version Information(default 0xFF)
Length	Number of bytes from COMMAND through to Check_LSB (typically 0x06)	Checksum not counted
CMD	Command byte	Means the specific operations, such as play / pause, etc.
Feedback	Command feedback	0x01: Feedback-send confirmation back to MCU; 0x00: No feedback
Param_MSB	Parameter	Most significant byte of parameter
Param_LSB	Parameter	Least significant byte of parameter
Check_MSB	Checksum	Most significant byte of checksum
Check_LSB	Checksum	Least significant byte of checksum
$O	End byte	0xEF

Ilustración 75. Formato de palabra del DFPlayer

En base a este formato, y tras encontrar una librería con todos los comandos, se escribieron dos funciones para construir la palabra a partir de los comandos previamente definidos en *fplayer.h*.

```
001  void putChr(uint8_t Word, uint8_t Indx, uint8_t Buff[])
002  {
003        Buff[Indx] = Word;
004  }
005
006  void buildCommand(uint8_t Word, uint16_t Param, uint8_t Buff[])
007  {
008        uint16_t xorsum = 0;
009        uint8_t i;
010
011        putChr(STARTBYTE, 0, Buff);
012        putChr(VERSIONBYTE, 1, Buff);
013        putChr(LENGTHBYTE, 2, Buff);
014        putChr(Word, 3, Buff);
015        putChr(FEEDBACKBYTE, 4, Buff);
016        putChr((uint8_t)(Param >> 8), 5, Buff);
```

```
017         putChr((uint8_t)(Param & 0x00ff), 6, Buff);
018
019         for(i=1;i<7;i++)
020         {
021               xorsum = xorsum + Buff[i];
022         }
023         xorsum = 0 - xorsum;
024
025         putChr((uint8_t)(xorsum >> 8), 7, Buff);
026         putChr((uint8_t)(xorsum & 0x00ff), 8, Buff);
027         putChr(ENDBYTE, 9, Buff);
028   }
```

La función *putChr* pone un byte en la posición Index del buffer. La función buildCommand utiliza la otra función para construir byte a byte la palabra completa. En el *main.c* se utiliza la función *HAL_UART_Transmit_IT* para mandar a través del puerto USART la palabra construida al módulo DFPlayer:

```
001       buildCommand(PLAYMP3SONG_CMD, 17, FPlayer_Buffer);
002       HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
```

Mediante el conversor USB-TTL y el terminal HERCULES se pudo corroborar que la trama de datos se enviaba correctamente (ilustración 77). Concretamente los comandos que se mandan en la imagen son los de reproducción de ficheros (mensajes o canciones) “001”, ”002”, “003”, y “004” de la carpeta *MP3* en la uSD.

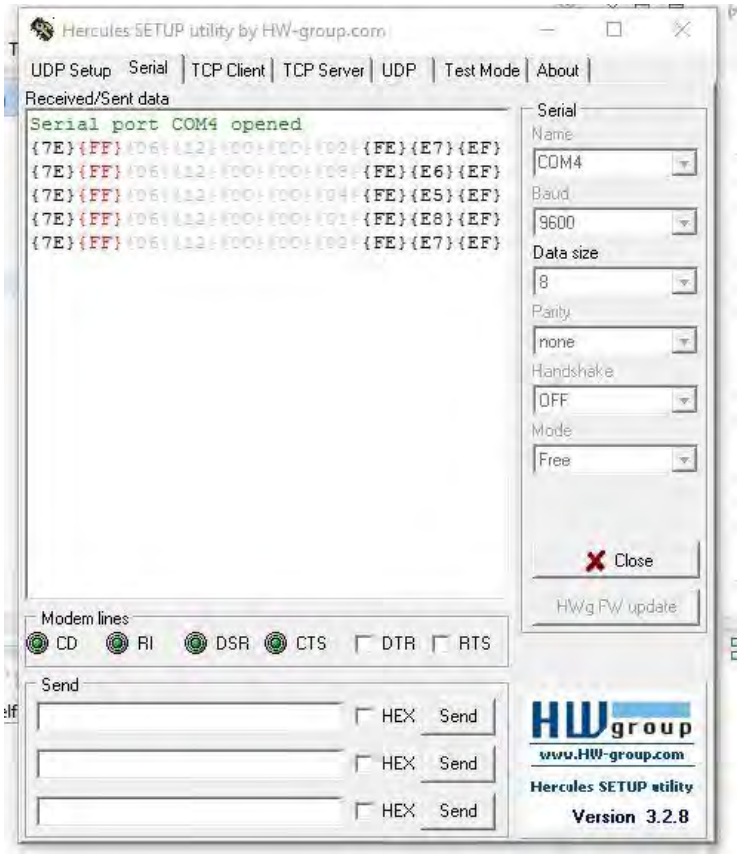


Ilustración 76. Comandos del DFPlayer impresos en el terminal HERCULES [imagen].

3.1.4. Receptor de Radiofrecuencia.

La implementación de la transmisión por radiofrecuencia vía puerto serie se estableció a través de un circuito mediante el cual se pulsan tres botones que van a las entradas del codificador RF600D (ilustración 78). La pulsación de estos botones produce cambios en las señales que entran al codificador. En función de dichos cambios, según lo expuesto en el apartado 2.3.4. se debería de modificar el byte 8 de la palabra que se envía.

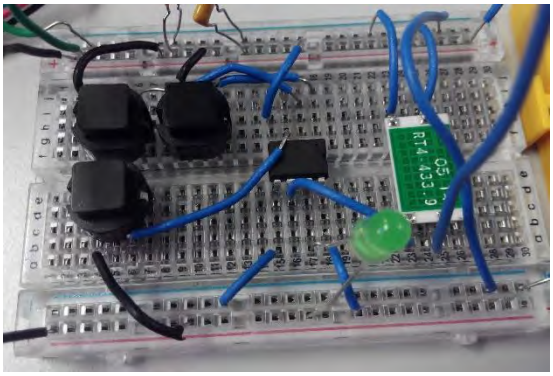


Ilustración 77. Sistema de transmisión por radiofrecuencia sobre placa de prototipado [Fotografía].

Utilizando el conversor TTL-USB y el terminal, se observó que efectivamente, estos cambios se producían (ilustración 79).

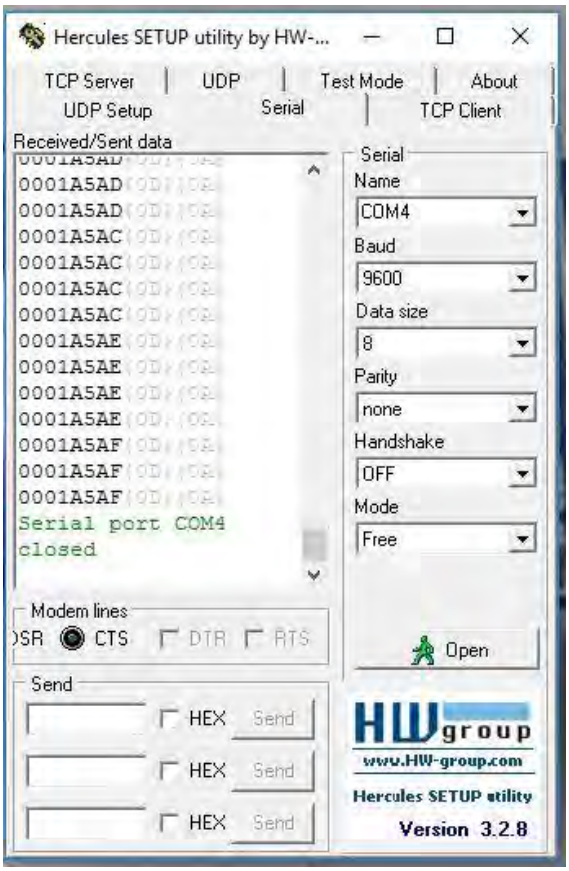


Ilustración 78. Trama de datos del receptor de radiofrecuencia RF600E [Imagen].

Sin embargo, esta trama se seguía transmitiendo una vez que se soltaba el botón. Por ello, con el objetivo de implementar en el programa principal una interpretación correcta de los datos que llegan del módulo receptor, se midió cada cuanto tiempo se producía esta transmisión. La conclusión es que la trama se envía cada 160 ms, como se puede observar en la ilustración 80.

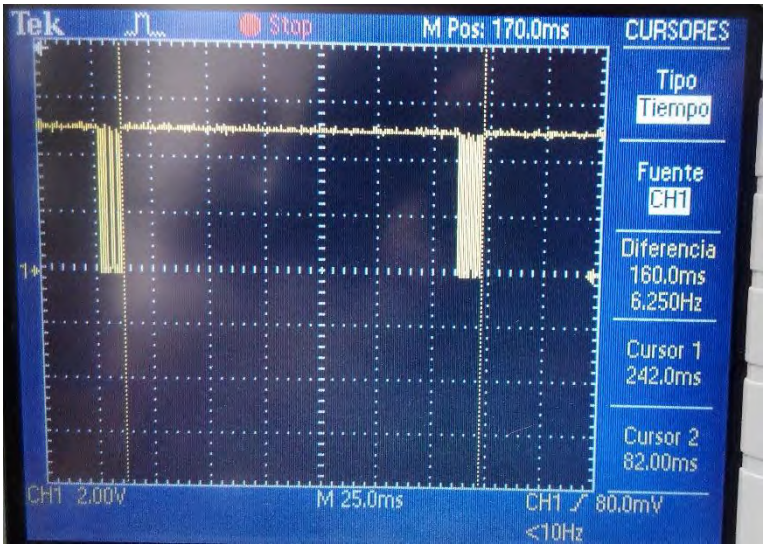


Ilustración 79. Intervalo de tiempo entre tramas serie del receptor de radiofrecuencia [Fotografía].

Cada vez que se envía una trama desde el Bloque Rampa al Bloque Carriles, se debe esperar al menos 160ms para que se haga efectivo un segundo envío, en el que el comando será '0', para que se deje de enviar esa trama:

```
001         RF_Command(commandToSend);
002         commandToSend = 0;
003         HAL_Delay(160);
004         RF_Command(commandToSend);
```

3.2. Firmware.

Para poder entender el funcionamiento del firmware del Bloque Carriles, se debe tener en cuenta que está condicionado en todo momento por los comandos que mediante radiofrecuencia se mandan desde el Bloque Rampa. Estos comandos son los que se muestra en la tabla de la ilustración 81.

		S3	S2	S1	S0	
0x00	Default	0	0	0	0	-
0x01	Player1	0	0	0	1	A
0x02	Player2	0	0	1	0	B
0x03	Player3	0	0	1	1	C
0x04	Player4	0	1	0	0	D
0x05	Character1	0	1	0	1	E
0x06	Character2	0	1	1	0	F
0x07	Character3	0	1	1	1	G
0x08	Character4	1	0	0	0	H
0x09	Points10	1	0	0	1	I
0x0A	Points15	1	0	1	0	J
0x0B	Points20	1	0	1	1	K
0x0C	Restart	1	1	0	0	L
0x0D	EndGame	1	1	0	1	M
0x0E	InitialPos	1	1	1	0	N
0x0F	NotUsed	1	1	1	1	O

Ilustración 80. Tabla de comandos de Radiofrecuencia [Tabla].

Algunos de estos comandos se mandan varias veces, por eso, el firmware del Bloque Carriles debe de estar preparado para recibir en cualquier momento una trama de datos y saber interpretarla. En la ilustración 82 se muestra un diagrama en el que se puede ver en qué momento del juego se envía, desde el Bloque Rampa, cada dato cuya letra corresponde al valor entre paréntesis.

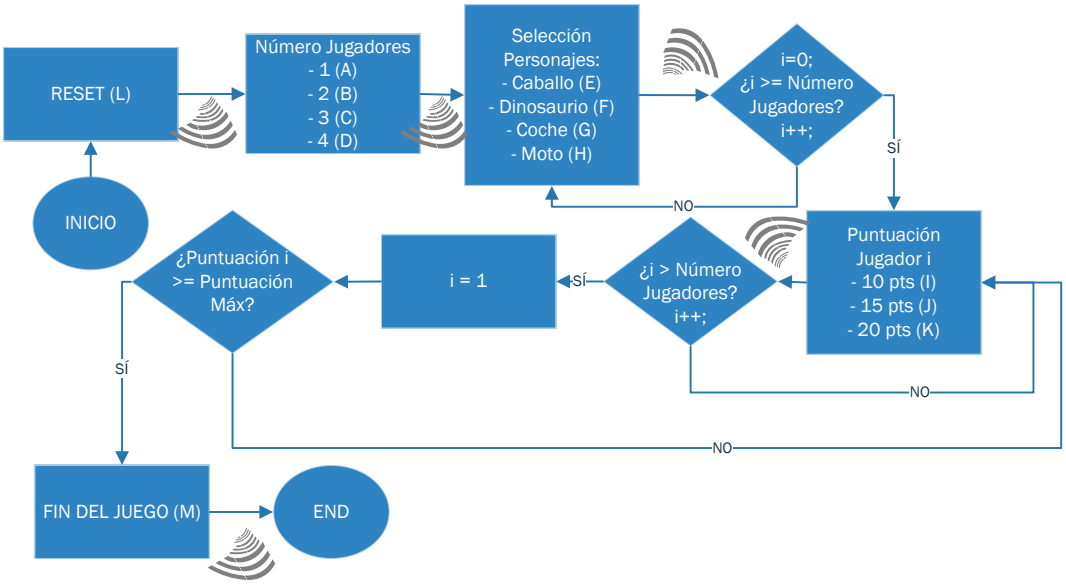


Ilustración 81. Diagrama que muestra en qué momento de la partida se envían los comandos RF [Diagrama].

3.2.1. Firmware del Bloque Carriles.

En las siguientes ilustraciones se muestra el diagrama de flujo del programa principal del Bloque Carriles. Posteriormente se explica en detalle cada parte del programa.

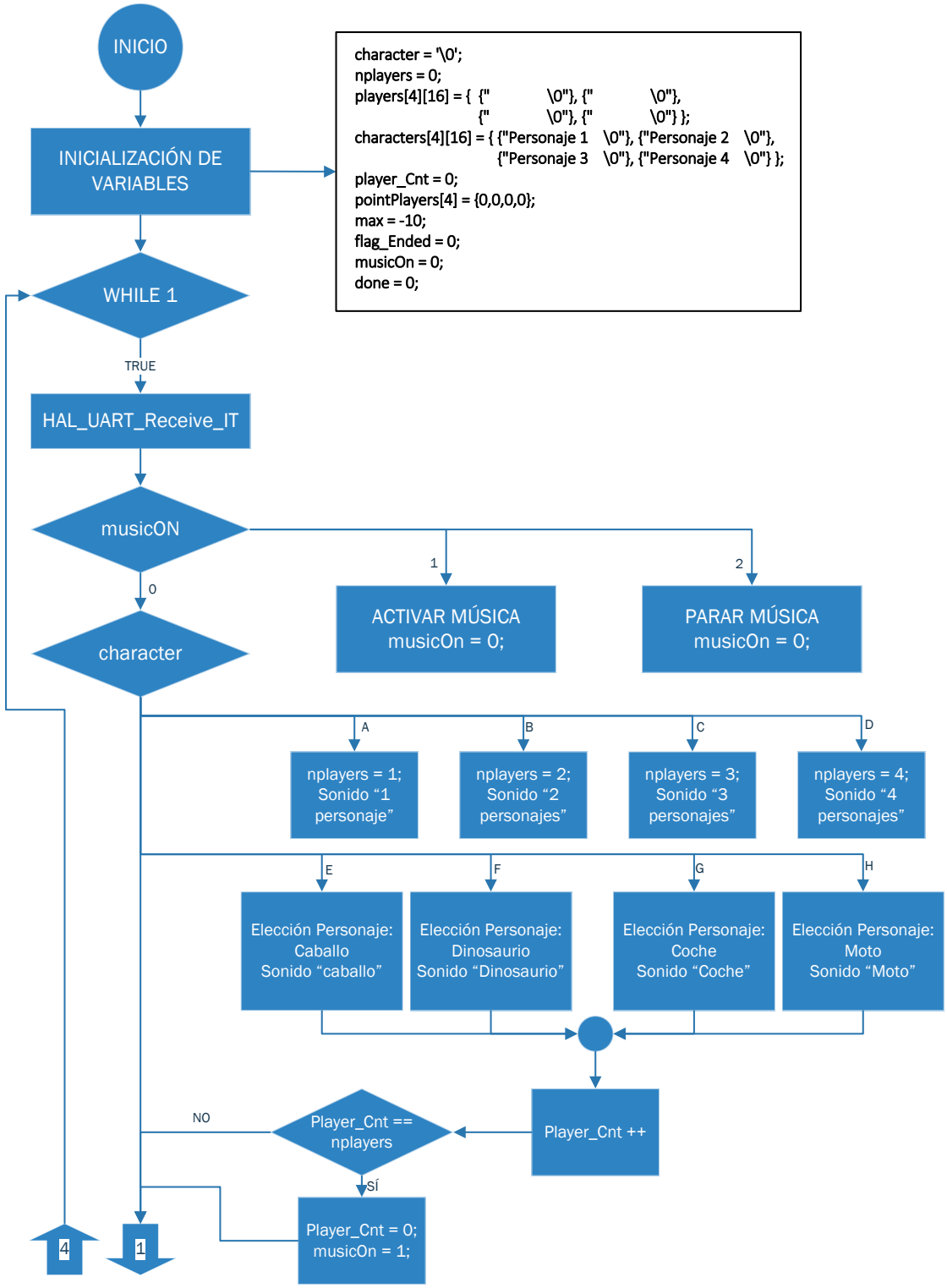


Ilustración 82. Diagrama de flujo del programa principal del Bloque Carriles. Parte 1 [Diagrama].

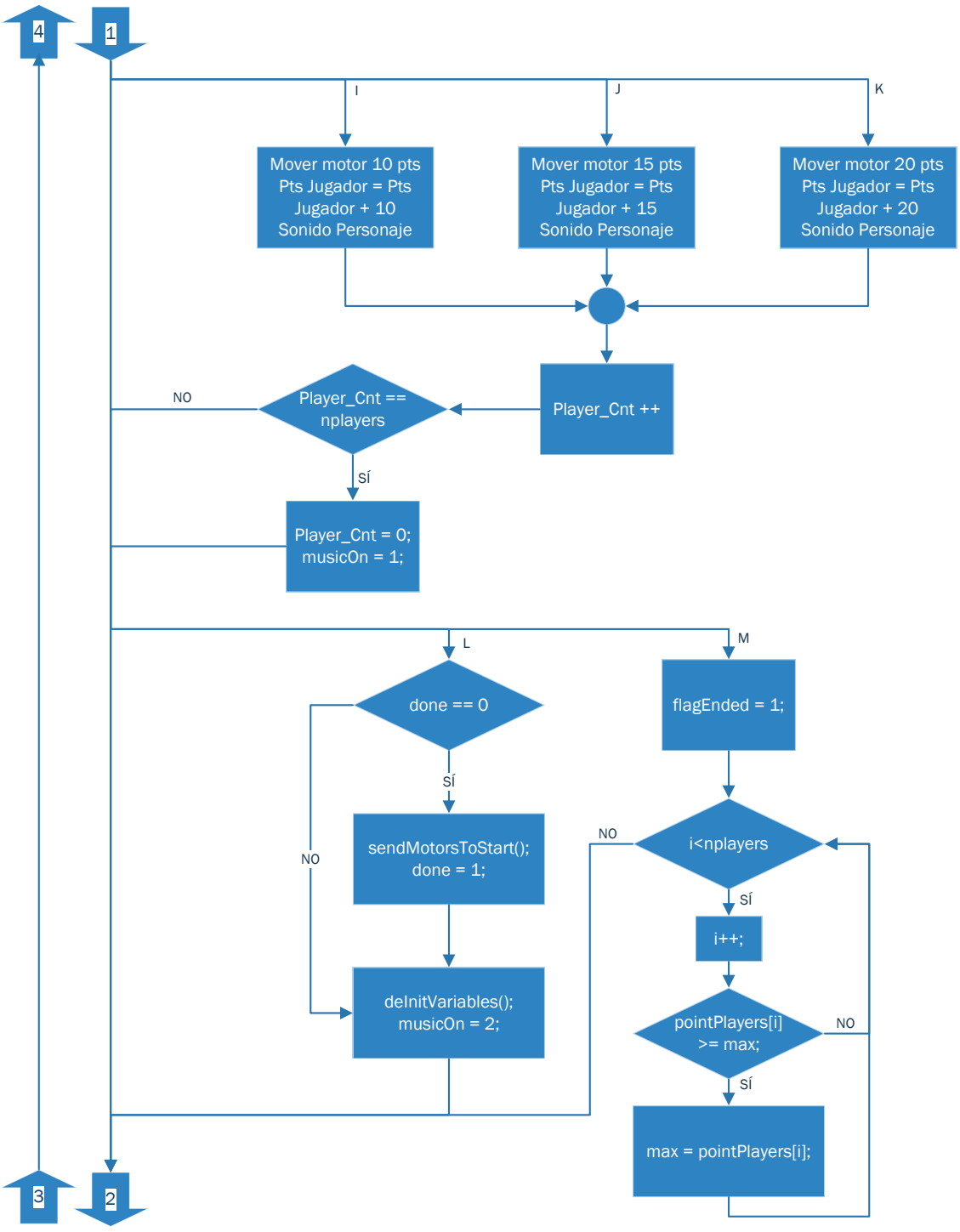


Ilustración 83. Diagrama de flujo del programa principal del Bloque Carriles. Parte 2 [Diagrama].

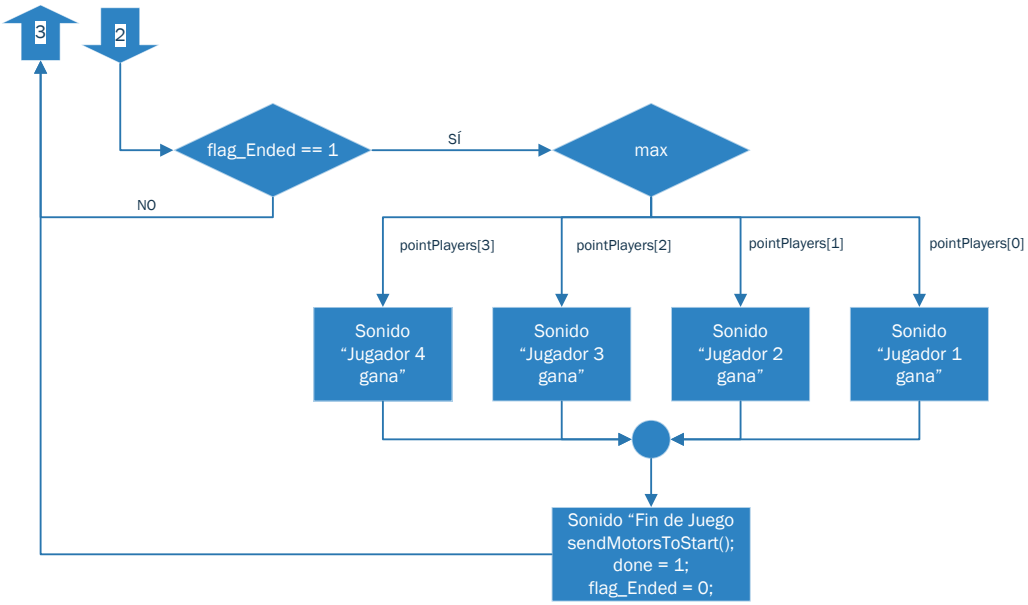


Ilustración 84. Diagrama de flujo del programa principal del Bloque Carriles. Parte 3 [Diagrama].

Nada más empezar el programa se inicializan las variables que se van a usar a lo largo de este. A continuación, y una vez que se ha entrado en el bucle *while*, se procede a la lectura de datos del puerto USART6, que es a donde llegan los datos procedentes del decodificador RF600D:

```
HAL_UART_Receive_IT(&huart6, RF_Buffer, LENGTH);
```

La función HAL_UART_Receive_IT recibe una cantidad de datos equivalente al tamaño de la palabra que se desee leer, en este caso *LENGTH = 10*. Cuando ha terminado el buffer (*RF_Buffer*) está lleno y preparado para ser manipulado.

A continuación, se chequea la variable *musicOn*. Si es 1 se activa la música de fondo, si es 2 se para, y si es 0 es porque ya ha sido activada y continúa sonando.

Después, dependiendo del carácter 8 del buffer *RF_Buffer* (según lo mencionado en apartado 2.3.4.) el programa reacciona de distintas formas:

- Si el carácter es 'A', 'B', 'C' o 'D' se asigna a la variable *nplayer* un número correspondiente al número de jugadores que vaya a haber en la partida. En función del número de jugadores, se manda también un comando para activar el correspondiente mensaje sonoro ("*Ha seleccionado: X jugadores*").

```
001         case 'A':
002             nplayers = 1;
003             buildCommand(PLAYMP3SONG_CMD, 01, FPlayer_Buffer);
004             HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
005             break;
```

- Si el carácter entrante es 'E', 'F', 'G' o 'H' se asigna a cada jugador un personaje distinto. La variable *player_Cnt* se irá incrementando cada vez que se seleccione un jugador para acceder a la cadena correspondiente. Cuando *player_Cnt* iguala a *nplayers* el siguiente paso es el comienzo del juego, por ello se cambia *musicOn* a 1 para que se active la música de fondo. Para guardar qué jugador eligió qué personaje, se utiliza la función *strncpy* y se copia la cadena del personaje elegido en la cadena del personaje que en ese momento haya seleccionado dicho personaje. Se manda también un comando para activar el correspondiente mensaje sonoro, que será un sonido característico del personaje escogido.

```
uint8_t players[4][16] =     {
                                        {"                  \0"},
                                        {"                  \0"},
                                        {"                  \0"},
                                        {"                  \0"}
                                  };
uint8_t characters[4][16] = {
                                        {"Personaje 1       \0"},
                                        {"Personaje 2       \0"},
                                        {"Personaje 3       \0"},
                                        {"Personaje 4       \0"}
                                   };
```

```
case'G':
            strncpy(players[player_Cnt], characters[2], 16);
            buildCommand(PLAYMP3SONG_CMD, 7, FPlayer_Buffer);
            HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
            break;
```

- El juego en este punto ya ha comenzado. Los caracteres a leer ahora pueden ser 'I', 'J' o 'K', que corresponden a haber conseguido 10, 15 o 20 puntos, respectivamente. En este momento ocurren cuatro cosas: se produce un sonido característico del personaje elegido por cada personaje, se produce un mensaje sonoro en función de la puntuación conseguida, se suma la puntuación conseguida y se mueve el motor del jugador correspondiente una distancia proporcional a dicha puntuación.

  Para conseguir lo primero es necesario ver qué personaje eligió cada jugador. Para ello se analiza el carácter 11 de la cadena de cada jugador, que corresponde al número de cada personaje (cada personaje tiene la forma de "*Personaje X \0*"). En función de ello se construye la palabra a enviar con *buildCommand* para reproducir un sonido u otro:

```
    if (players[player_Cnt - 1][10] == '1')
    {
            buildCommand(PLAYMP3SONG_CMD, 9, FPlayer_Buffer);
    }

    if (players[player_Cnt - 1][10] == '2')
```

```
007     {
008             buildCommand(PLAYMP3SONG_CMD, 10, FPlayer_Buffer);
009     }
010
011     if (players[player_Cnt - 1][10] == '3')
012     {
013             buildCommand(PLAYMP3SONG_CMD, 11, FPlayer_Buffer);
014     }
015
016     if (players[player_Cnt - 1][10] == '4')
017     {
018             buildCommand(PLAYMP3SONG_CMD, 12, FPlayer_Buffer);
019     }
```

Al igual que en el apartado anterior, la variable *player_Cnt* se incrementa para que se repita el proceso con el jugador siguiente. Este proceso se repetirá hasta que desde el Bloque Rampa se alcance la máxima puntuación por cualquiera de los jugadores.

- Cuando uno cualquiera de los jugadores llega al final, se manda el comando con la letra 'M'. Se activa el flag que indicala finalización del juego *flag_Ended = 1* y se chequea qué jugador fue el que consiguió la mayor puntuación, esa puntuación se almacena en la variable *max*.

```
001     if(character == 'M')
002     {
003             flag_Ended = 1;
004             for (i=0 ; i<nplayers ; i++)
005             {
006                             if (pointPlayers[i] >= max)
007                             {
008                                             max = pointPlayers[i];
009                             }
010             }
011     }
```

Cuando *flag_Ended* es igual a 1, se produce un mensaje sonoro indicando qué jugador ha ganado. Por último, se mandan uno a uno todos los motores al inicio al mismo tiempo que se activa una música de victoria. Se activa una variable llamada *done* para indicar que los motores ya han sido llevados a sus posiciones iniciales.

```
001         if (flag_Ended == 1)
002         {
003               if (max == pointPlayers[3])
004               {
005                     buildCommand(PLAYMP3SONG_CMD, 16, FPlayer_Buffer);
006                     HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
007               }
008               else if (max == pointPlayers[2])
009               {
010                     buildCommand(PLAYMP3SONG_CMD, 15, FPlayer_Buffer);
011                     HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
012               }
013               else if (max == pointPlayers[1])
```

```
                {
                      buildCommand(PLAYMP3SONG_CMD, 14, FPlayer_Buffer);
                      HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
                }
                else if (max == pointPlayers[0])
                {
                      buildCommand(PLAYMP3SONG_CMD, 13, FPlayer_Buffer);
                      HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
                }

                HAL_Delay(3000);
                buildCommand(PLAYMP3SONG_CMD, 21, FPlayer_Buffer);
                HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
                sendMotorsToStart();
                done = 1;
                flag_Ended = 0;
          }
```

- Por último, el juego está diseñado para que en cualquier momento de la partida se pueda reiniciar la misma. Cuando esto sucede se manda desde el Bloque Rampa el comando con la letra 'L'. Dentro de esta parte del programa se reinicializan todas las variables y, si la variable *done* está a 0, se envían los motores a su posición inicial. La varable *musicOn* se pone a 2 para parar la música en caso de que hubiera alguna en ese momento.

```
if(character == 'L')
{
                        if (done == 0)
                        {
                                              sendMotorsToStart();
                                              done = 1;
                        }
                        deInitVariables();
                        musicOn = 2;
}
```

A partir de este momento el programa, sin la necesidad de resetear el microcontrolador ni de apagar y encender la fuente de alimentación, es capaz de volver a interpretar los datos procedentes del Bloque Rampa para empezar un nuevo juego.

El programa completo del Bloque Carriles está disponible en los anexos.

3.3. Fabricación y ensamblaje.

Para tener el sistema completamente terminado y listo para funcionar, lo único que falta es el ensamblaje de las placas de circuito impreso, la integración de estas en los chasis, y la fabricación de los cables y componentes necesarios para la interacción de los chasis con las placas.

El ensamblaje de las PCB's se llevó a cabo teniendo en cuenta las necesidades descritas en el apartado 2.4. En las siguientes ilustraciones se muestra este proceso.

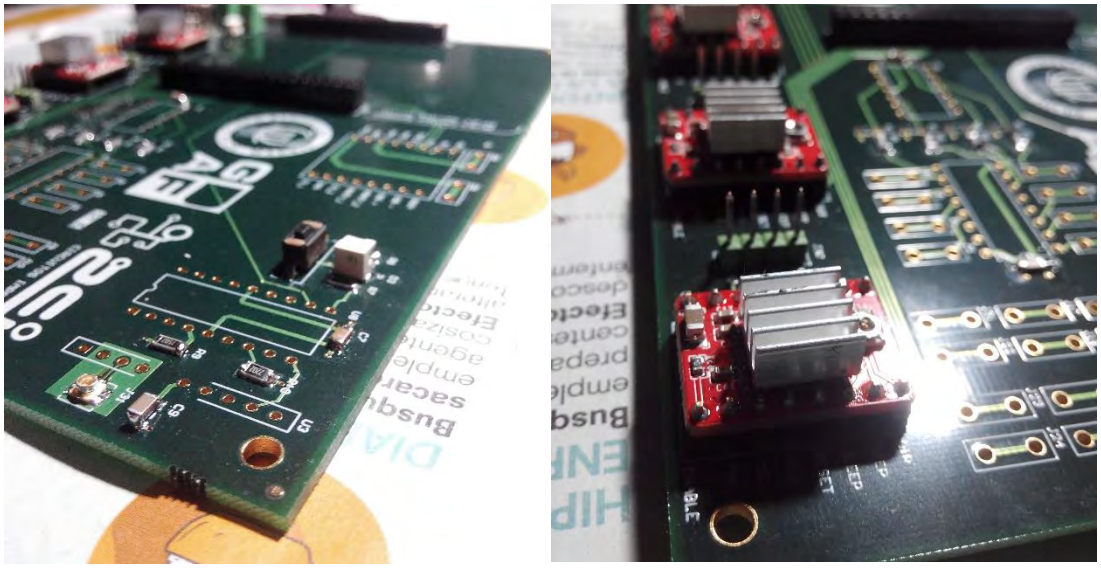


Ilustración 85. Proceso de ensamblaje de las placas de circuito impreso 1 [Fotografía].

En primer lugar, se colocaron todos los componentes cuya dificultad de soldar era mayor, como los conectores U-FL o los componentes de montaje superficial (leds, resistencias, condensadores…). Después se soldaron los conectores de los microcontroladores y los conectores de alimentación. Se continuó con los componentes de orificio pasante (módulos StepStick, integrados, módulos de radiofrecuencia, etc.). Por último, se soldaron todos los conectores metálicos de la serie Faston 250, que es a donde van conectados la mayoría de los cables.

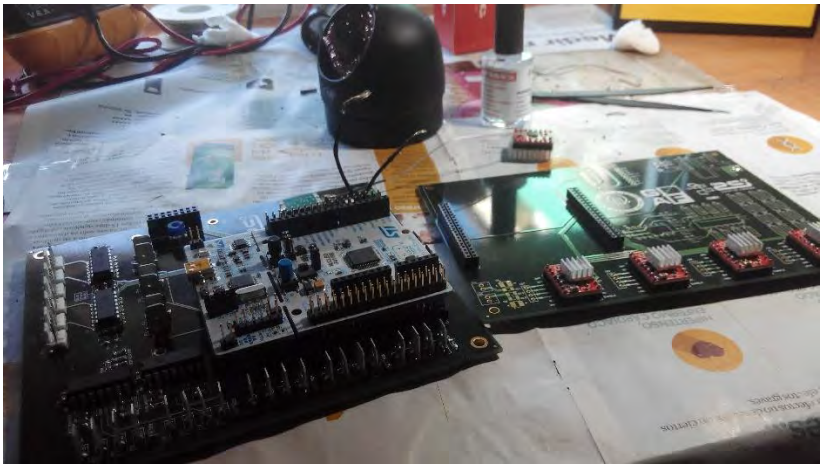


Ilustración 86. Proceso de ensamblaje de las placas de circuito impreso 2 [Fotografía].

El montaje final es el que se muestra en la ilustración 88:

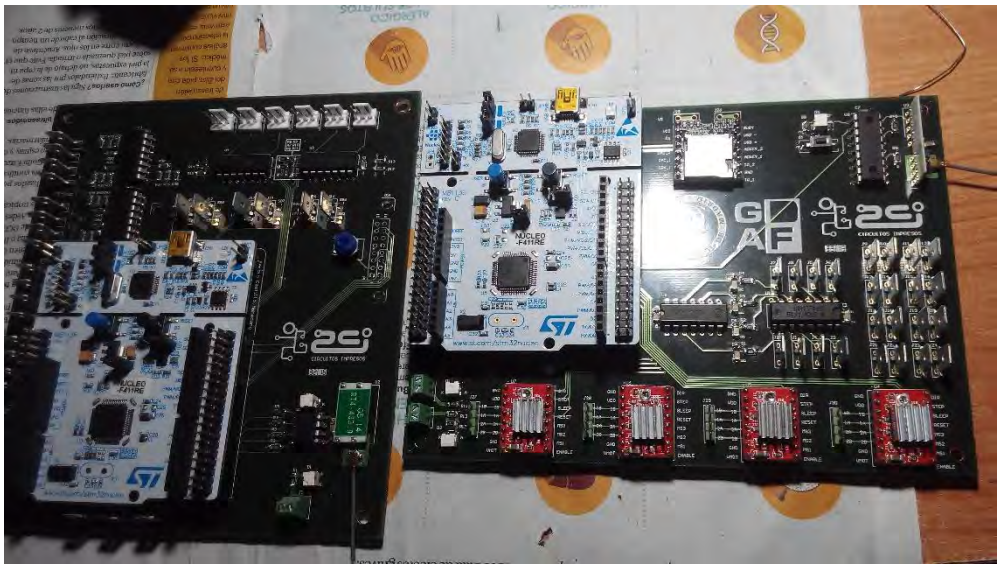


Ilustración 87. Proceso de ensamblaje de las placas de circuito impreso 3 [Fotografía].

Cabe destacar que, para hacer operativo el botón RESET del panel de control del Bloque Rampa, se decidió soldar mediante hilo de *wrapping* los terminales del botón a los puntos de soldadura del botón reset de la placa de desarrollo del microcontrolador (ilustraciones 89 y 90).

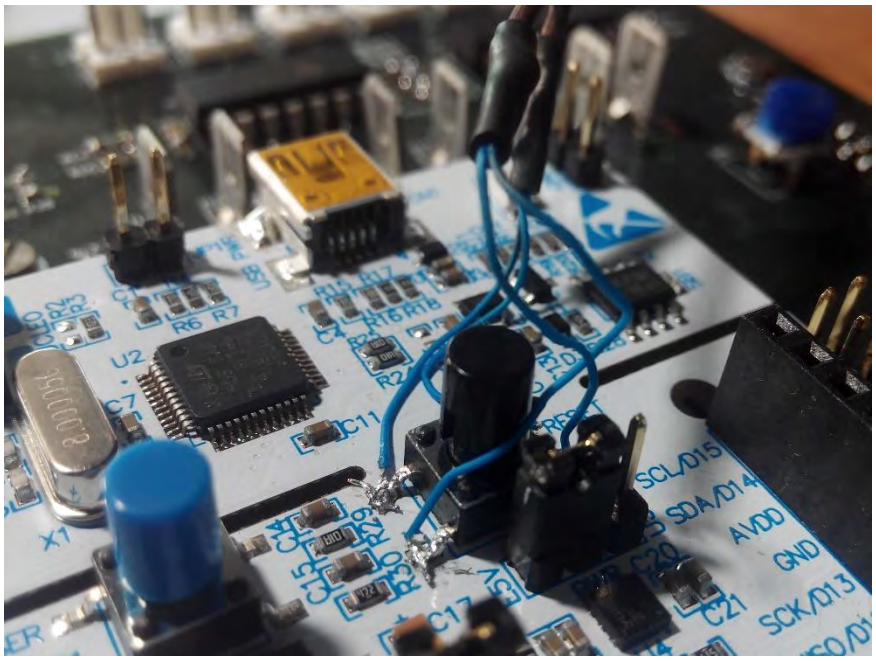


Ilustración 88. Proceso de ensamblaje de las placas de circuito impreso 4 [Fotografía].

Ilustración 89. Proceso de ensamblaje de las placas de circuito impreso 5 [Fotografía].

Para la fabricación de todos los cables se siguió un procedimiento similar:

- Medir longitud aproximada del cable y pelarlo.
- Introducir un trozo de funda termorretráctil, después el conector aéreo hembra en el cable y soldar.
- Calentar funda termorretráctil sobre punto de soldadura.

Todos los cables tienen un aspecto parecido al de la ilustración 91.

Ilustración 90. Cable soldado a conector con funda termorretráctil [Fotografía].

En la ilustración siguiente se muestra el aspecto de la placa del Bloque Rampa con todo el cableado conectado.

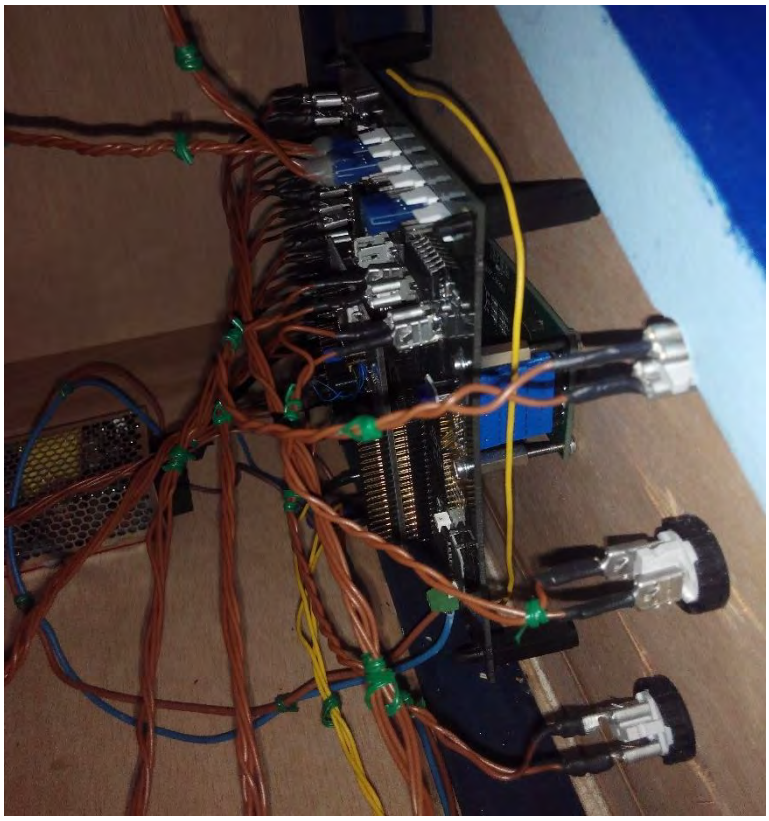


Ilustración 91. Conexionado del cableado de la placa de Bloque Rampa [Fotografía].

4. Capítulo 4. Pruebas y resultados experimentales.

4.1. Pruebas.

Aunque la mayor parte del programa se desarrolló y se probó previamente en las placas de circuito impreso antes de ser integradas en el chasis. Una vez que estas se integraron hubo que cambiar ciertos aspectos del programa para facilitar el uso del juego lo máximo posible. Por otro lado, también se tuvieron que hacer modificaciones sobre la placa debido a cambios de diseño. A continuación, se describen algunos de los problemas principales de los que se tuvieron constancia mientras se realizaban pruebas.

- Bloqueo de los motores 2 y 3

Apareció la primera vez que se procedió a mover los motores a través de la caída de la pelota sobre las casillas de la rampa. Se tenía que solo se movían los motores 1 y 4. El problema estaba en la forma en que se actuaba sobre los motores a través de los flags de codificador 8:3 de los microrruptores. El programa nunca entraba en la instrucción que hacía el *toggle* sobre el correspondiente pin. Se solucionó cambiando la forma en que los flags eran leídos, añadiendo una variable local llamada *stop*.

- Error al leer la pulsación de los microrruptores (1)

Según la tabla de verdad de la ilustración 32, la entrada de mayor peso (la 7) que no se usa debe estar a 5V para que las salidas cambien en función del resto de las entradas. Sin embargo, durante el diseño de la PCB este pin fue llevado a tierra en lugar de a 5V, como consecuencia de ello las salidas del codificador 8:3 nunca cambiaba cualquiera que fuera el valor de sus entradas. Para solucionar este problema se desmontó la placa del chasis y se procedió a llevar, mediante cable de *wrapping*, el correspondiente pin a 5V (ilustración 93). Para ello, tal como se ve en la ilustración, hubo que "arrancar la patilla de la PCB y asegurarse de que no hacía contacto con el *pad*.

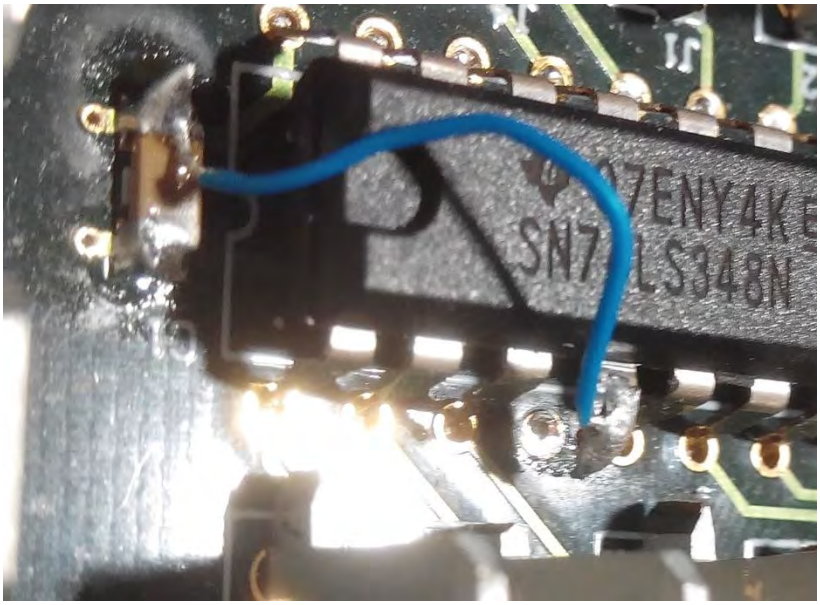


Ilustración 92. Modificación en el diseño de la PCB. Pata del codificador 8:3 llevada a 5V a través de hilo *wrapping* [Fotografía].

- Error al leer la pulsación de los microrruptores (2)

En el apartado 2.3.2 de la descripción de la electrónica asociada a los microrruptores, se menciona la necesidad de añadir unas resistencias de descarga justo a la entrada de los amplificadores operacionales para permitir el paso de la corriente por ellas.

Sin embargo, y a pesar de que se tuvo en cuenta durante la fase de prototipado, durante el diseño de la PCB este detalle se pasó por alto. Como consecuencia, cuando se pulsaba el microrruptor una vez, el motor sí que reaccionaba y se paraba, pero cuando se volvía a pulsar, como el nivel de tensión no había variado, el flanco de subida que debería producirse para ser detectado por el microcontrolador no se producía y por tanto el motor no se paraba. Para solucionarlo se desmontó de nuevo la placa del chasis y se procedió a soldar en la parte inferior de la misma tres resistencias de orificio pasante de 1 kohm en los correspondientes pines (ilustración 94).

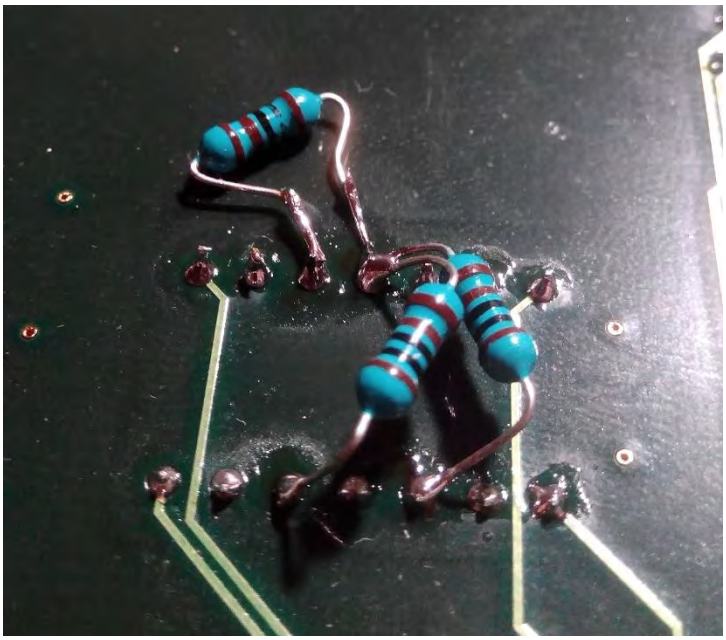


Ilustración 93. Modificación en el diseño de la PCB. Resistencias de orificio pasante para habilitar el paso de la corriente por ellas [Fotografía].

- Modificación en la interfaz de accionamiento de trampilla de la rampa.

El botón mediante el que se produce el accionamiento de la apertura de la rampa es un relé que tiene sus tres terminales conectados a un conector Jack macho *TRS*[11] de 3.5mm. Sin embargo, el tipo de botón más usado en el Colegio San Rafael es un Jack *TS*[12]. Es decir, mientras allí se usa con preferencia un conector de dos terminales, el juego se diseñó para uno de tres. En las reuniones iniciales se produjo un fallo de comunicación del tipo concreto de Jack.

[11] TRS: Tip-Ring-Sleeve
[12] TS: Tip-Sleeve

Según las ilustraciones 95 y 96, que muestran el esquema eléctrico de la interfaz de nuestro botón con el microcontrolador: en estado de reposo el camino que está cerrado es el que se forma entre *sleeve* y *ring* (GND y entrada a microcontrolador). Cuando se produce la pulsación el camino que se cierra es el que forman *tip* y *ring* (3,3V y GND).

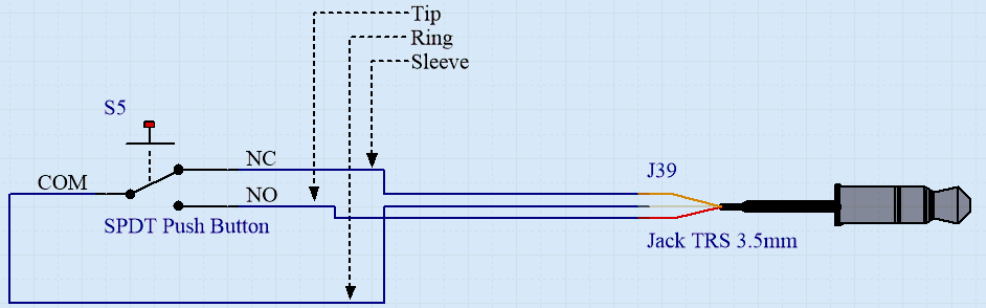


Ilustración 94. Interfaz de conexión del botón con el microcontrolador (1) [Esquema].

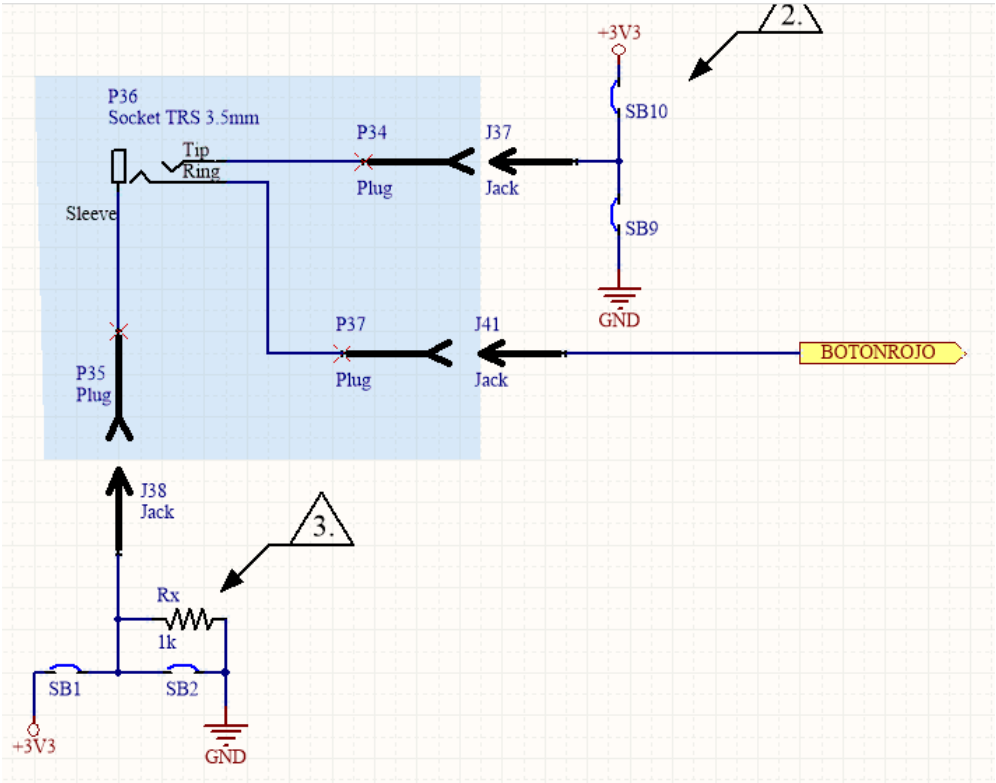


Ilustración 95. Interfaz de conexión del botón con el microcontrolador (2) [Esquema].

Si se conectara un Jack *TS* a la interfaz tal y como se ha descrito, se produciría un corto entre el *sleeve* y el *ring* debido a que el terminal *sleeve* en los conectores *TS* son más largos (ilustración 97).

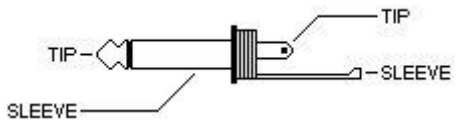


Ilustración 96. Conector TS de 3.5mm [Esquema]. *(28)*

En estado de reposo no pasaría nada, pero si se pulsara y se cerrara el camino entre *tip* y *ring*, se produciría un cortocircuito entre *tip* (3,3V), *ring* y *sleeve* (GND). Para evitarlo, se tuvo que desmontar la placa del chasis, desoldar el punto de soldado *SB2* y colocar una resistencia de 1kohm en paralelo (ilustración 98).

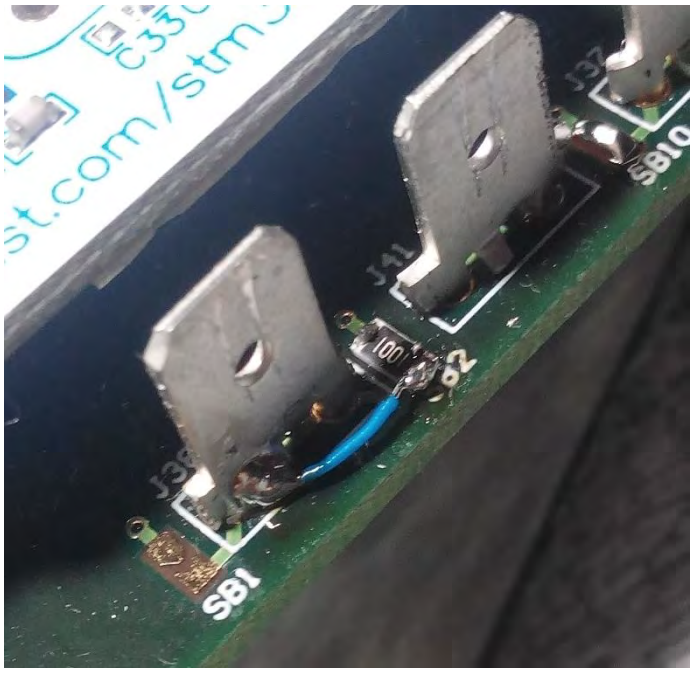


Ilustración 97. Modificación en el diseño de la PCB. Resistencias de montaje superficial colocada en paralelo al punto de soldado SB2 [Fotografía].

4.2. Pruebas de campo.

Durante las pruebas realizadas en el colegio, todos los problemas que hubieron surgido con anterioridad, descritos en el apartado anterior, no aparecieron. El botón funcionó a la perfección y los chicos y chicas del colegio pudieron disfrutar de la partida hasta el final.

El único problema fue relativo a la pelota de pingpong. No hubo problemas en pruebas previas, pero allí había ocasiones en las que el sensor infrarrojo no detectaba su paso. No obstante, sí que detectaba el paso de la mano. Como solución se les ha planteado a los monitores ajustar la cabeza de los leds para que estén lo mejor alineado posible con el receptor, ya que el haz que lanza es muy direccional. Una vez se hizo esto, el paso de la bola ya era detectado con normalidad.

Así todo, al recoger la pelota de la casilla, el paso de la mano era detectado, por lo que la partida seguía transcurriendo con total normalidad.

A continuación, se deja el enlace de dos vídeos, en los que se puede ver una prueba en el laboratorio, y el transcurso normal de la partida en el colegio:

Prueba Juego Carreras Adaptado (1): https://www.youtube.com/watch?v=pyFjskidAT8

Prueba Juego Carreras Adaptado (2): https://www.youtube.com/watch?v=5OllvHny_DY

5. Conclusiones, posibles mejoras y líneas futuras.

5.1. Conclusiones.

La realización de este proyecto siempre estuvo enfocada a la consecución de los objetivos propuestos al principio, que fueron cumplidos. Sin embargo, más allá de lograr el correcto funcionamiento del juego, el propósito a nivel personal fue siempre la aplicación de conocimientos teóricos y el aprendizaje de herramientas que se usan en el ámbito profesional. De ahí la razón (entre otras) de no haber planteado en ningún momento una placa Arduino, y haber propuesto la utilización de algo más orientado al ámbito profesional.

Por ello, el autor de la presente memoria queda doblemente satisfecho. Y si además tenemos en cuenta el noble propósito al que está dirigido el proyecto, no queda más que dar las gracias por estos meses de trabajo y sacrificio, que aunque difíciles, han merecido enormemente la pena para todas las partes implicadas.

5.2. Mejoras.

De las mejoras a nivel mecánico que se podrían haber aplicado en el proyecto se habla en la memoria de Sergio Aguilera Sevilla.

Con respecto al firmware y a la electrónica, hay dos mejoras en especial que merecen la pena ser mencionadas ya que hubieran supuesto un ahorro de tiempo y recursos:

Por un lado, y aunque no suele ser lo habitual, en este microcontrolador la mayoría de los pines, cuando están configurados como entradas de propósito general (GPIO Input) son tolerantes a niveles de tensión de 5V, por lo que en el caso de los microrruptores (por ejemplo), mediante la elección correcta de los pines que tienen esta característica, se podría haber ahorrado la etapa de acondicionamiento de la señal en la que se reduce el nivel de tensión de 5V a 3,3V (es decir, los comparadores).

Por otro lado, en lo que se refiere a los servomotores, si en lugar de haber configurado las salidas en modo *push-pull* se hubieran configurado en modo *open-drain*, y habiendo añadido una resistencia de *pull-up* a un nivel de tensión de 5V a la salida de cada pin de control *PWM*, se habrían conseguido los 5V de nivel alto de tensión sin la necesidad de los comparadores que en el actual diseño sirven para tal efecto.

5.3. Líneas futuras.

Con respecto a mejoras que se podrían aplicar sobre el diseño actual, hay una especialmente significativa en la cual se pensó desde el principio. La elección de personajes está limitada a tan solo 4. Sin embargo, sería muy sencillo poder utilizar más personajes si se deseara.

Una vez se tuviera otro personaje con una base preparada para ser colocado sobre los carriles, en el firmware del Bloque Rampa tan solo habría que añadir a la cadena que almacena los personajes una fila más con el nombre del personaje en cuestión para que aparezca en el LCD a principio de la partida. En el Bloque Carriles habría que hacer lo mismo y además almacenar los sonidos correspondientes al personaje en la uSD, la cual se puede extraer fácilmente. De esta forma, cada jugador podría elegir el personaje que más le gustara.

6. Presupuesto.

Código	Ud.	Descripción	Medición	P.U.	P.T.
01		**Materiales y construcción mecánica**			
01.01	2	**Madera de calabo** Tablero de madera de 250x100x7 cm.	Ud.	32,42	64,84
01.02	8	**Listones madera** Listones de DM de 100x3x3 cm.	Ud.	1,50	12
01.03	1	**Bobina de Plástico PLA** Filamento de plástico PLA de 1.75mm. 1Kg.	Ud.	19,95	19,95
01.04	3	**Pintura** Pintura al agua de distintos colores. Bote de 500ml.	Ud.	4,50	13,50
01.05	1	**Bobina de Aluminio** Bobina de aluminio 1 mm diámetro.	Ud.	16,52	16,52
01.06	1	Pegamento Pack barras de pegamento termoplástico.	Ud.	3	3
01.07	1	**Pegamento** - Cola blanca rayt500gr 429-07	Ud.	2,60	2,60
01.08	1	**Bridas** Bolsa con 10 unidades grandes.	Ud.	0,60	0,60
01.10	4	**Alcayatas**	Ud.	0,15	0,60
01.11	1	**Espigas** Bolsa con 100 unidades.	Ud.	1	1
01.12	4	**Personajes** Juguetes distintos de plástico para usarlos de personajes.	Ud.	2	8
01.13	4	**Rodamientos** Rodamiento de bolas RS Pro, Miniatura, 625-2Z.	Ud.	1,43	5,72
01.14	1	**Imanes** Tiras magnéticas 100mm PP, Adhesivo trasero, 2.3mm de grosor, 10u.	Ud.	5,56	5,56
01.15	1	**Varilla** Varilla roscada M5 1metro+8tuercas M5 autoblocantes+4 tornillos M5X20 (para sujetar poleas conducidas).	Ud.	1,8	1,80
01.16	8	**Tuerca** Tuercas auto-freanantes de M5	Ud.	0,1	0,80
01.17	1	**Manguitos** Manguitos metálicos M5-M5 Hembra-Hembra 8 unidades para sujetar poleas conducidas.	Ud.	1	1
01.18		**Tornillería** - 16 Tornillos M3X20 y 16 Tuercas M3 (para los apoyos intermedios de los carriles). - 16 Tornillos M3X50 (4 por motor). - Pack Tornillos (para la construcción de las estructuras).	Ud.	1,28 1,6 2	4.88
		TOTAL:			**162.37**

Código	Ud.	Descripción	Medición	P.U.	P.T.
02	**Componentística común a los dos Bloques**				
02.01	51	**63824-1** TE Connectivity Serie FASTON 250 Terminal desconexión rápida hembra PCB	Ud.	0,0754	3.83
02.02	51	**60838-1** TE Connectivity Serie FASTON 250 Terminal desconexión rápida macho aéreo	Ud.	0,0744	3,79
02.03	5	**LYA67K-J2M1-26** Led SMD Amarillo 2Ma SMD	Ud.	0,1020	0,51
02.04	2	**RNF-100-3/32-0-STK** Tubo Termo retráctil, Flexible, Ignífugo, 2.4 mm, 0.094 ", 2:1, Negro, 3.94 ft, 1.22 m	Ud.	1,16	2,32
02.05	8	**501R18W103KV4E** Condensadores de desacoplo, SMD 1206 0.1uF	Ud.	0,0861	0,69
02.06	24	**MC0125W120611K** Res 1206 SMD, 1kohm, 200V, 1206, 125mW, ± 1%	Ud.	0,0034	0,08
02.07	19	**MCWR12X1502FTL** Res 1206 SMD, 15kohm, 200V, 125mW	Ud.	0,0121	0,23
02.08	9	**MCWR12X1000FTL** Res 1206 SMD, 100ohm, 200V, 125mW	Ud.	0,0118	0,11
02.09	5	**MC0125W1206122K** Res 1206 SMD, 22kohm, 200V, 125mW	Ud.	0.0036	0,2
02.10	2	**PCB** Placa de circuito estándar FR-4 de cuatro capas.	Ud.	0	0
		TOTAL:			**11,76**

Código	Ud.	Descripción	Medición	P.U.	P.T.
03	**Componentes Bloque Rampa**				
03.01		Microcontrolador			
03.01.01	1	**STM32F411-RE** STM CORTEX-M4 MCU Placa de desarrollo	Ud.	9,97	9,97
03.01.02	4	**06.84.256** Soporte PCB para montaje en panel M4 ETTINGER	Ud.	0,523	2,09
03.02		Zona de detección			
03.02.01	10	**OP550C** OPTEK TECHNOLOGY Fototransistor	Ud.	0,662	6,62
03.02.02	10	**TSAL6200** VISHAY Emisor de Infrarrojos, Alta Potencia, 100mA	Ud.	0,289	2,89
03.02.03	2	**LM234n** Amplificador Operacional	Ud.	0,526	1,05
03.03		Radiofrecuencia			
03.03.01	1	**RF600E** RF SOLUTIONS IC, ENCODER KEELOQ, DIP8	Ud.	3,63	3,63
03.03.02	1	**AM-RT4-433FR** RF SOLUTIONS RF/AM MODULE TRANSMITTER 433MHZ	Ud.	8,41	8,41
03.04		Servomotores			
03.04.01	1	**SG-90** Servomotor 9g a pequeña escala, 200mA	Ud.	8,32	8,32
03.04.02	2	**LM234n** Amplificador Operacional	Ud.	0,526	1,05
03.04.03	6	**640454-3**	Ud.	0,0935	0,56

		TE Connectivity conector IDC cable a placa, 3p 2.54mm			
03.04.04	6	**3-640442-3** receptáculo cable a placa 3p 2.54mm	Ud.	0,117	0,70
03.05		Circuito de Acondicionamiento			
03.05.01	1	**Botón Dome Rojo** Botón Pulsador BrikoGeeks	Ud.	19,24	19,24
03.05.02	1	**AUDL-73-2M** Cable 2m con terminación Jack trs pelado	Ud.	2,576	2,58
03.05.03		**KLB 4** Conector Jack de montaje en chasis	Ud.	1,24	1,24
03.06		Panel de control			
03.06.01	3	**1.10.001.011** Botón demontaje en panel, (ON)-OFF Rafi	Ud.	2,69	8,07
03.06.02	1	**MC21605B6WR-BNMLW** MIDAS - Pantalla lcd 16x02	Ud.	11,06	11,06
03.06.03	1	**416PA203M** Potenciómetro 20kohm	Ud.	0,65	0,65
03.06.04	1	**2214S-16SG-85** 16x socket 2.54mm agujero pasante	Ud.	1,39	1,39
03.06.05	1	**2213S-16G** 16x header 2.54mm agujero pasante	Ud.	0,416	0,42
03.06.06	4	**R25-1001402** Separador latón hembra hex m2.5	Ud.	0,365	1,46
03.07		Alimentación			
03.07.01	1	**TXM 025-105** Fuente de Alimentación Cerrada AC/DC, Compacta, Ajustable, Fijo, 90 V, 264 V, 25 W, 5 V, 5 A	Ud.	18,8	18,8
03.07.02	1	**MC34231-091-72** Interruptor basculante para montaje en panel spst	Ud.	1,20	1,20
		TOTAL:			**111,85**

Código	**Ud.**	**Descripción**	**Medición**	**P.U.**	**P.T.**
04	**Bloque Carriles**				
04.01		Microcontrolador			
04.01.01	1	**STM32F411-RE** STM CORTEX-M4 MCU Placa de desarrollo	Ud.	9,97	9,97
04.01.02	4	**06.84.256** Soporte base adhesiva ETTINGER	Ud.	0,341	1,36
04.02		Motores paso a paso			
04.02.01	1	**42BYGHW811** Cuatro motores paso a paso bipolares NEMA 17	Ud.	59	59
04.02.02	4	**StepStick** Módulo A4988 de Reprap basado en driver A4988	Ud.	5,95	23,8
04.03		Finales de carrera			
04.03.01	8	**D2FD-01L30-1H** Microrruptor SPDT OMRON ELECTRONICS	Ud.	1,078	8,62
04.03.02	1	**SN74LS348N** Encoder 8:3 TEXAS INSTRUMENTS, 16DIP	Ud.	5,48	5,48
04.04		Sistema de Audio			
04.04.01	2	**ABS-230-RC** altavoz 4ohm 4w	Ud.	4,97	9,94

04.04.02	1	**DFPlayer** Módulo Driver de audio MP3 con zócalo para uSD	Ud.	7,99	7,99
04.05		Radiofrecuencia			
04.05.01	1	**AM-HRR30-433** RF SOLUTIONS RECEIVER AM	Ud.	5,04	5,04
04.05.02	1	**RF600D** RF SOLUTIONS IC, DECODER KEELOQ	Ud.	5,41	5,41
04.06		Alimentación			
04.06.01	1	**TXL 060-0512DI** Fuente de Alimentación AC/DC Cubierta, Compacta, Ajustable, Fija, 5V, 8A, 12V, 4A	Ud.	57,62	57,62
04.06.02	1	**MC34231-091-72** Interruptor basculante para montaje en panel spst	Ud.	1,20	1,20
		TOTAL:			**195.43**

El coste de desarrollo e implementación del sistema por parte de los dos ingenieros a cargo del proyecto, tomando como referencia el salario medio de un ingeniero titulado de la Universidad Carlos III de Madrid de 26,34€/hora desglosado en:

- Sueldo neto por hora de un ingeniero junior de 20,5€.
- Un 28% para la Seguridad Social.
- 1,5% por desempleo

Habiendo empleado un total de 480 horas durante 6 meses, con una dedicación de 4 horas al día, 20 días a mes:

$$\frac{26{,}34€}{hora} \cdot 480\ horas \cdot 2\ ingenieros = 25.286{,}4\ €$$

APARTADO	*PRECIO*
Materiales y construcción mecánica	162,37 €
Material electrónico común a los dos Bloques	11,76 €
Bloque Rampa	111,82 €
Bloque Carriles	195,43 €
Coste de dos ingenieros	25.286,4 €
COSTE TOTAL	**25.667,78 €**

Bibliografía

1. **Koynos.** Centro de Educación Especial Koynos. *Koynos Cooperativa Valenciana.* [En línea] 11 de Septiembre de 2016. http://koynos.org/es/centro-de-educacion-especial/transicion-a-la-vida-adulta.html.

2. **Alcázar Álvarez, Juan Antonio.** ardilladigital. *Ardilla Digital. Recursos Digitales para Educación Especial.* [En línea] Noviember de 2009. [Citado el: 11 de Septiembre de 2016.] http://ardilladigital.com/DOCUMENTOS/EDUCACION%20ESPECIAL/TVA/La%20transicion%20a%20la%20vida%20adulta%20-%20Alvarez%20-%20art.pdf.

3. **Boletín Oficial del Estado.** Gobierno de España. *Agenda Estatal.* [En línea] 2003. [Citado el: 12 de Septiembre de 2016.] https://www.boe.es/buscar/act.php?id=BOE-A-2003-22066&p=20131203&tn=0.

4. **Lourdes Moreno y Paloma Martínez.** Open Course Ware. *OCW - UC3M.* [En línea] [Citado el: 12 de Septiembre de 2016.] http://ocw.uc3m.es/ingenieria-informatica/evitando-barreras-accesibilidad/material-de-clase-en-formato-pdf/TEMA2_disenouniversal.pdf.

5. **Colegio de Educación Especial San Rafael.** [En línea] [Citado el: 12 de Septiembre de 2019.] http://www.sanrafaelcolegio.com/.

6. **Portal Aragonés de la Comunicación Aumentativa y Alternativa.** ARASAAC. *Gobierno de Aragón.* [En línea] [Citado el: 12 de Septiembre de 2016.] http://arasaac.org/aac.php.

7. **Parque de Bolas.** parquedebolas. [En línea] [Citado el: 12 de Septiembre de 2016.] http://www.parquedebolas.com/es/productos/categoria/39/atracciones-para-ferias-y-fiestas-carrera-de-camellos-caballos.

8. **Ceapat Imserso.** slideshare. *AIJU.* [En línea] [Citado el: 12 de Septiembre de 2016.] http://es.slideshare.net/ceapatimserso/juegojuguetes-ydiscapacidad-la-importancia-del-diseo-universal.

9. **Brikogeek.** [En línea] [Citado el: 2016 de Septiembre de 14.] http://tienda.bricogeek.com/componentes/162-boton-dome-rojo.html.

10. **Condit, Reston.** Microchip. [En línea] [Citado el: 2016 de Septiembre de 15.] http://homepage.cs.uiowa.edu/~jones/step/an907a.pdf.

11. **RepRap.** RepRap. *StepStick Bipolar Stepper Motor Driver.* [En línea] [Citado el: 15 de Septiembre de 2016.] http://reprap.org/wiki/Stepstick.

12. **Allegro MicroSystems.** Datasheet. *A4988 Bipolar Stepper Motor Driver.* [En línea] [Citado el: 15 de Septiembre de 2016.] https://www.pololu.com/file/0J450/A4988.pdf.

13. **Texas Instruments.** SN74LS348N. *8:3 Encoder, Switching Characteristics Table - Page 4.* [En línea] [Citado el: 23 de Septiembre de 2016.] http://www.ti.com/lit/ds/symlink/sn74ls348.pdf.

14. **Texas Intruments.** LM324N. *6.5 Electrical Characteristics.* [En línea] [Citado el: 23 de Septiembre de 2016.] http://www.ti.com/lit/ds/symlink/lm124-n.pdf.

15. **DFRobot.** *Drive the Future.* [En línea] [Citado el: 16 de Septiembre de 2016.] https://www.dfrobot.com/index.php?route=product/product&product_id=1121&search=dfplayer.

16. **Farnell.** [En línea]

17. **STMicroelectronics.** ST. [En línea] [Citado el: 16 de Septiembre de 2016.] http://www.st.com/content/ccc/resource/technical/document/application_note/05/fb/41/91/39/02/4d/1e/CD00259245.pdf/files/CD00259245.pdf/jcr:content/translations/en.CD00259245.pdf.

18. —. ST. *STM32100B-EVAL demonstration firmware.* [En línea] [Citado el: 16 de Septiembre de 2016.] http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32-standard-peripheral-libraries-expansions/stsw-stm32040.html.

19. **Wikipedia.** The Free Encyclopedia. *Codificación Manchester.* [En línea] [Citado el: 16 de Septiembre de 2016.] https://es.wikipedia.org/wiki/Codificaci%C3%B3n_Manchester.

20. **RFSolutions.** RF600E/RF600D. [En línea] [Citado el: 16 de Septiembre de 2016.] http://www.farnell.com/datasheets/56726.pdf.

21. **Wikipedia.** The Free Encyclopedia. *Antena Marconi.* [En línea] [Citado el: 16 de Septiembre de 2016.] https://es.wikipedia.org/wiki/Antena_Marconi.

22. **Farnell.** element14. *RF600D.* [En línea] [Citado el: 16 de Septiembre de 2016.] http://es.farnell.com/rf-solutions/rf600d/ic-decoder-keeloq-dip18/dp/1200973.

23. **Lorente, Alba Rodriguez.** Desarrollo de una ayuda Técnica para alumnos del colegio San Rafael (11). *Mando y panel multisensorial interactivo (1).* Leganés : Universidad Carlos III de Madrid, 2014.

24. **Wikipedia.** The Free Encyclopedia. *Resina Epoxy.* [En línea] [Citado el: 17 de Septiembre de 2016.] https://es.wikipedia.org/wiki/Resina_epoxi.

25. **IPC-4412A.** Apendix II Finished Fabric Glass Styles. *Table II-1 Finished Fabric Glass Styles in SI Units.* [En línea] [Citado el: 2016 de Septiembre de 2016.] https://www.ipc.org/4.0_Knowledge/4.1_Standards/IPC-4412A-Amends-1-2-3_12-17-10.pdf.

26. **2CISA.** Circuitos Impresos. [En línea] [Citado el: 17 de Septiembre de 2016.] http://www.2cisa.com/.

27. **Wikipedia.** The Free Encyclopedia. *Impedancia de entreada.* [En línea] [Citado el: 17 de Septiembre de 2016.] https://es.wikipedia.org/wiki/Impedancia_de_entrada.

28. **Leads Direct.** The Power To Connect. *What do TS and TRS mean?* [En línea] [Citado el: 20 de Septiembre de 2016.] http://www.leadsdirect.co.uk/knowledge-base/what-do-ts-and-trs-mean/.

29. **Wikipedia.** The free encyclopedia. [En línea] [Citado el: 13 de Septiembre de 2016.] https://es.wikipedia.org/wiki/Reduced_instruction_set_computing.

30. **AIJU.** Instituto Tecnológico de producto infantil y ocio. [En línea] [Citado el: 14 de Septiembre de 2016.] http://www.aiju.info/.

31. **Escudero, Miguel De Prado.** UC3M. [En línea] [Citado el: 14 de Septiembre de 2016.] https://docs.google.com/viewerng/viewer?url=e-archivo.uc3m.es/bitstream/handle/10016/20320/TFG_Miguel_De-Prado_Escudero.pdf.

32. **QuimiNet.** QuimiNet. *Informacón y Negocios segundo a segundo.* [En línea] 12 de Enero de 2016. [Citado el: 2016 de Septiembre de 15.] https://www.quiminet.com/articulos/acrilonitrilo-butadieno-estireno-abs-descripcion-propiedades-y-aplicaciones-4433.htm.

33. **Wikipedia.** The Free Encyclopedia. [En línea] [Citado el: 13 de Septiembre de 2016.] https://es.wikipedia.org/wiki/M%C3%A1quina_de_Galton.

34. **stm32duino.** DFPlayer Mini Extended Library for STM32. [En línea] [Citado el: 18 de Septiembre de 2016.] http://www.stm32duino.com/viewtopic.php?t=405.

Anexos.

Placa de Circuito Impreso del Bloque Carriles 101
Esquema Eléctrico 101
Plano de Situación de Componentes 109
Plano mecánico 111
Placa de Circuito Impreso del Bloque Rampa 122
Esquema Eléctrico 122
Plano de Situación de Componentes 131
Plano mecánico 133
Firmware Bloque Carriles 144
main.c 144
fplayer.h 150
fplayer.c 152
steppermotors.h 153
stm32f4xx_it.c 157
Manual de Usuario 159

DIBUJA	FECHA
Sergio Castillo	08/09/2016

NOTAS:

Aquellos componentes en azul no pertenecen a la placa, si no a un nivel superior. Se muestran en los esquemas con tal de facilitar la comprensión de los circuitos.

1. Cambio realizado a posteriori sobre la placa. Se ha llevado el pin 7 del integrado U1 hasta 5V. Inicialmente a GND.

2. Cambio realizado a posteriori sobre la placa. Se han soldado tres resistencias de orificio pasante de 1kohm y 0.25w desde los pines 3, 5 y 10 del integrado U9 a GND.

3. Cambio realizado a posteriori sobre la placa. Condensador C9 se coloca entre pines 1 y 2 del integrado U6. Inicialmente estaba entre pin 6 y GND (en cortocircuito ahora).

SISTEMA CARRILES

HOJA	DESCRIPCIÓN
1	ÍNDICE
2	GLOBAL
3	MICROCONTROLADOR
4	ALIMENTACIÓN
5	MOTORES PAP
6	MICRORRUPTORES
7	AUDIO
8	RADIOFRECUENCIA

Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

Autores
Sergio Castillo Mohedano
Sergio Aguilera Sevilla

PROYECTO
PCB Carriles

TÍTULO
Índice

ESCALA / SCALE	FORMATO / SIZE	HOJA / SHEET
1/1	A4	1 / 8

DIBUJA	FECHA
Sergio Castillo	08/09/2016


Logo GDAF

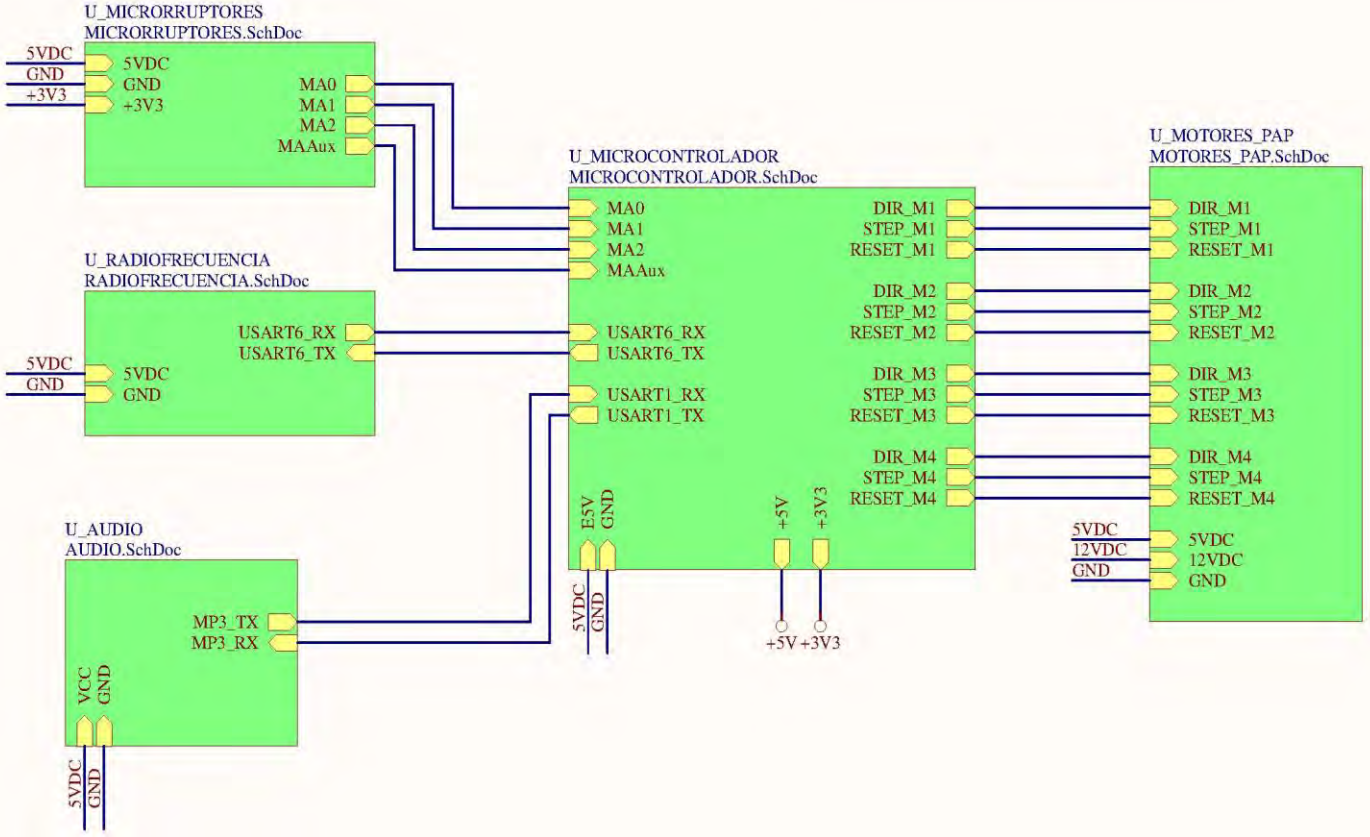

Logo UC3M

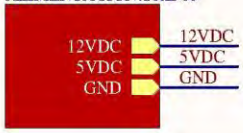

Logo 2CISA

FD1 FD3 FD2 FD4

MH1 DRILL 3.2 mm PTH
MH3 DRILL 3.2 mm PTH
MH2 DRILL 3.2 mm PTH
MH4 DRILL 3.2 mm PTH

H1 SPACER_ADHESIVE_M3
H3 SPACER_ADHESIVE_M3
H2 SPACER_ADHESIVE_M3
H4 SPACER_ADHESIVE_M3





Universidad Carlos III de Madrid

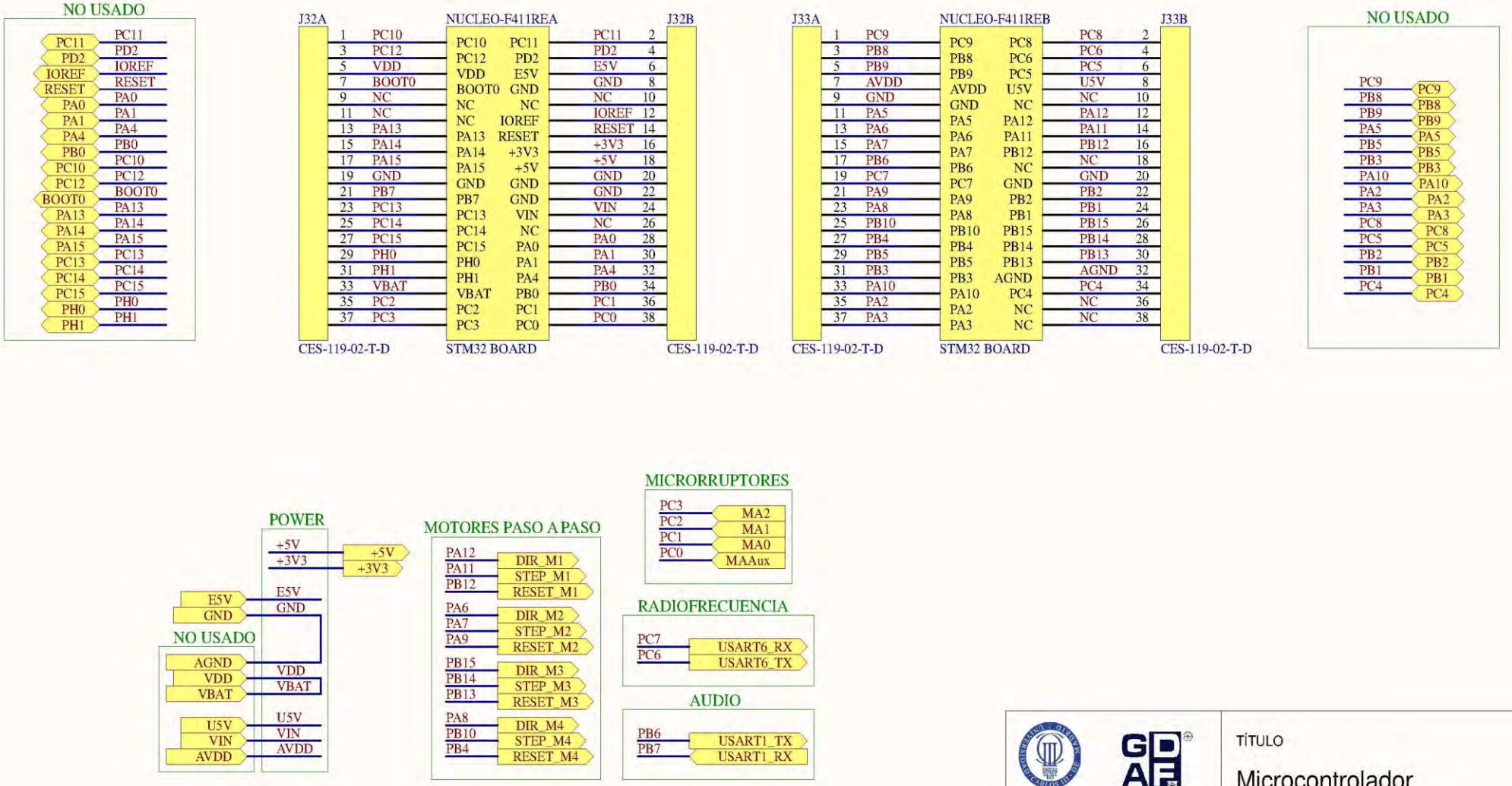
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Esquema Global

ESCALA / SCALE	FORMATO / SIZE	HOJA / SHEET
1/1	A4	2 / 8

DIBUJA	FECHA
Sergio Castillo	08/09/2016



Universidad Carlos III de Madrid

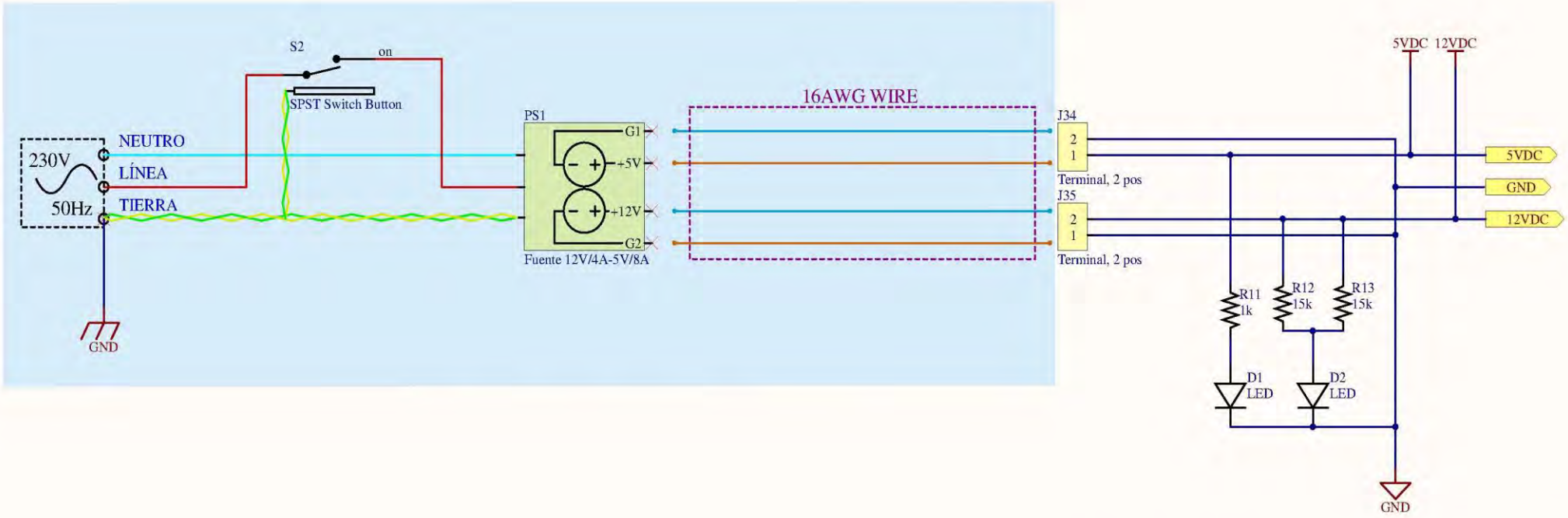
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Microcontrolador

ESCALA / SCALE	FORMATO / SIZE	HOJA / SHEET
1/1	A4	3 / 8

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Alimentación

ESCALA SCALE	FORMATO SIZE	HOJA SHEET
1/1	A4	4 / 8

DIBUJA	FECHA
Sergio Castillo	09/09/2016

12VDC
5VDC
GND

DIR_M1
STEP_M1
RESET_M1
DIR_M2
STEP_M2
RESET_M2
DIR_M3
STEP_M3
RESET_M3
DIR_M4
STEP_M4
RESET_M4

U7
U6
U5
U4

DIR
STEP
SLEEP
RESET
MS3
MS2
MS1
ENABLE
GND
VDD
1B
1A
2A
2B
VMOT
5VDC
12VDC

A4988 StepStick

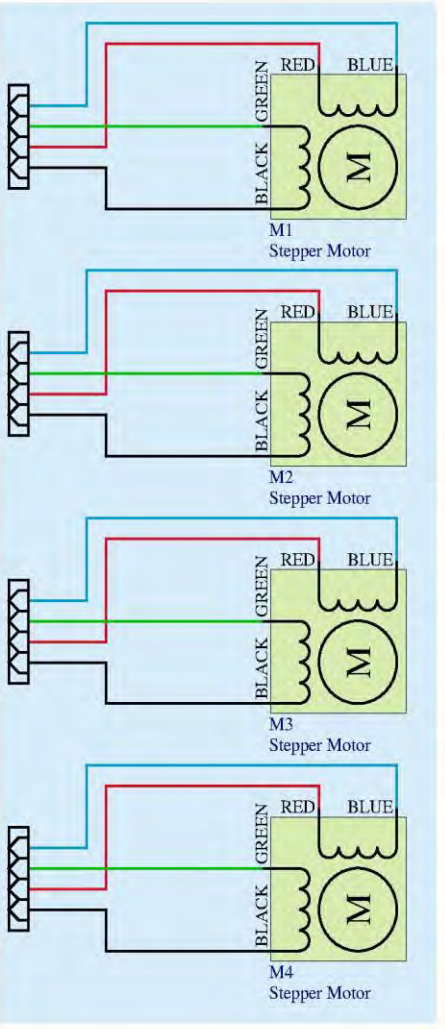
J27
J28
J29
J30

PIN HEADER 4X

M1
M2
M3
M4

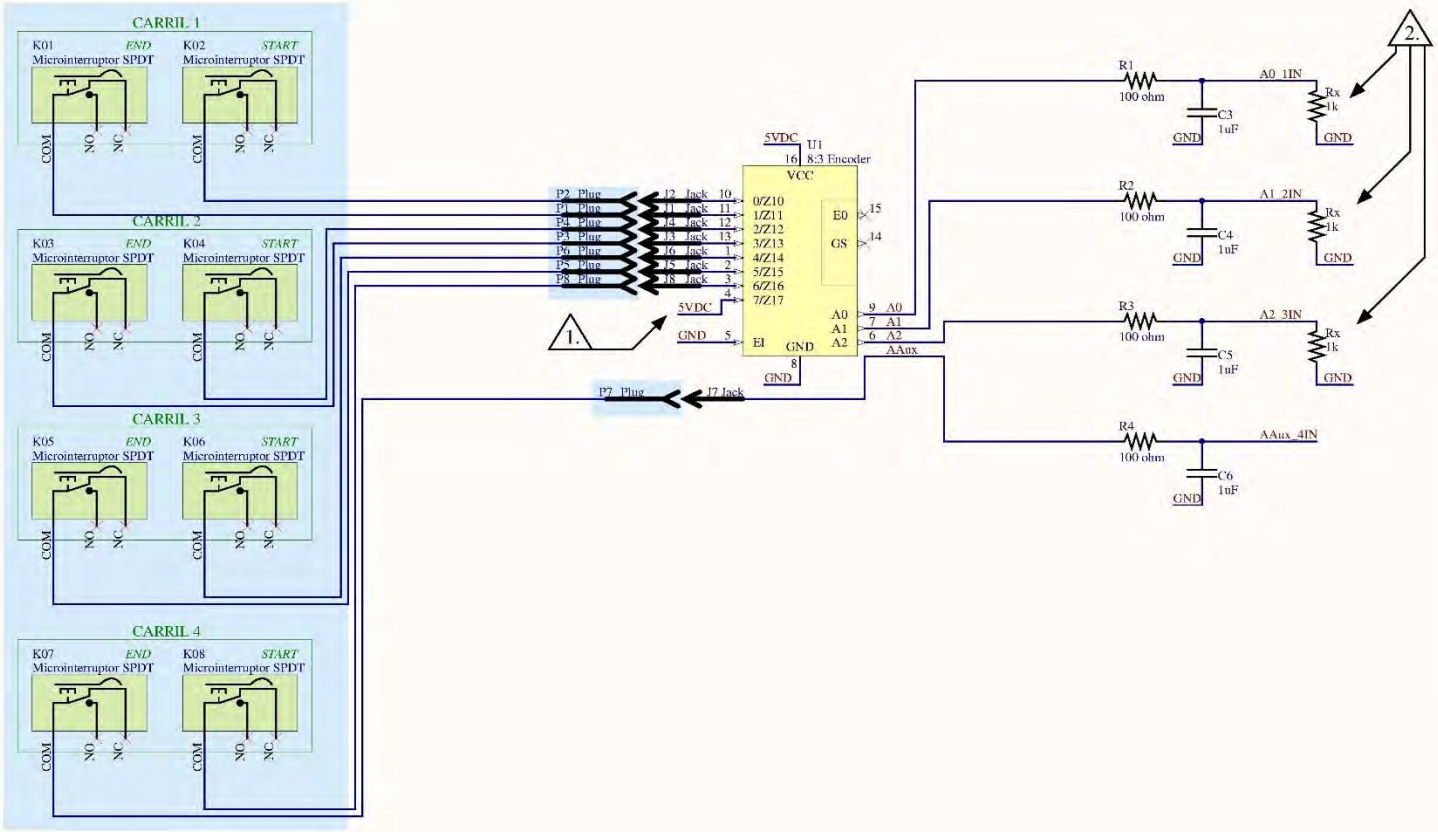
Stepper Motor

RED
BLUE
GREEN
BLACK



Universidad Carlos III de Madrid	TÍTULO: Motores Paso a Paso
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.	ESCALA / SCALE 1/1 · FORMATO / SIZE A4 · HOJA / SHEET 5 / 8

DIBUJA	FECHA
Sergio Castillo	09/09/2016



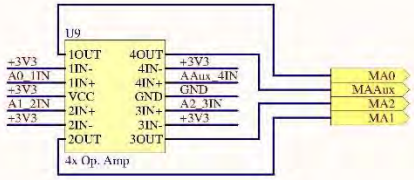
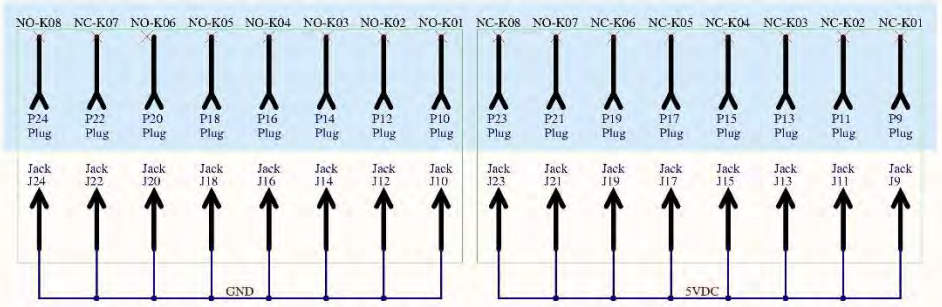


TABLA ENCODER

	EI	0	1	2	3	4	5	6	7	A2	A1	A0	Aaux
	1	X	X	X	X	X	X	X	X	Z	Z	Z	-
	0	1	1	1	1	1	1	1	1	Z	Z	Z	-
	0	X	X	X	X	X	X	X	0	0	0	0	-
N0-K08	0	X	X	X	X	X	X	0	1	0	0	1	-
N0-K05	0	X	X	X	X	X	0	1	1	0	1	0	-
N0-K06	0	X	X	X	X	0	1	1	1	0	1	1	-
N0-K03	0	X	X	X	0	1	1	1	1	1	0	0	-
N0-K04	0	X	X	0	1	1	1	1	1	1	0	1	-
N0-K01	0	X	0	1	1	1	1	1	1	1	1	0	-
N0-K02	0	0	1	1	1	1	1	1	1	1	1	1	-
N0-K07	X	X	X	X	X	X	X	X	X	-	-	-	1

Condensadores de desacoplo



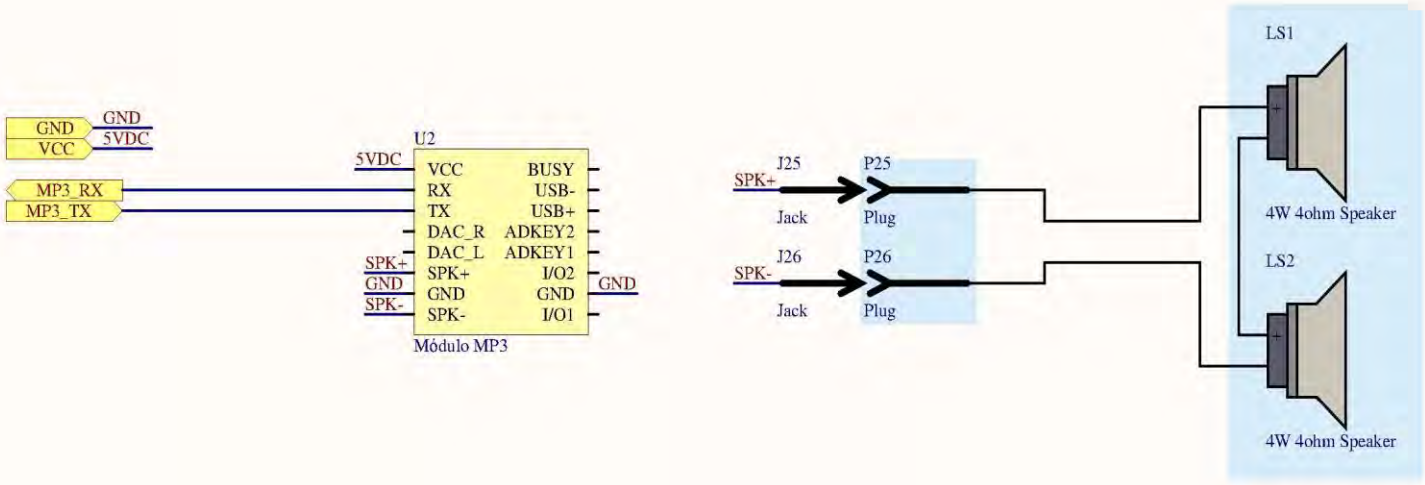
Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Microrruptores

ESCALA SCALE	1/1	FORMATO SIZE	A3	HOJA SHEET	6 / 8



DIBUJA	FECHA
Sergio Castillo	09/09/2016

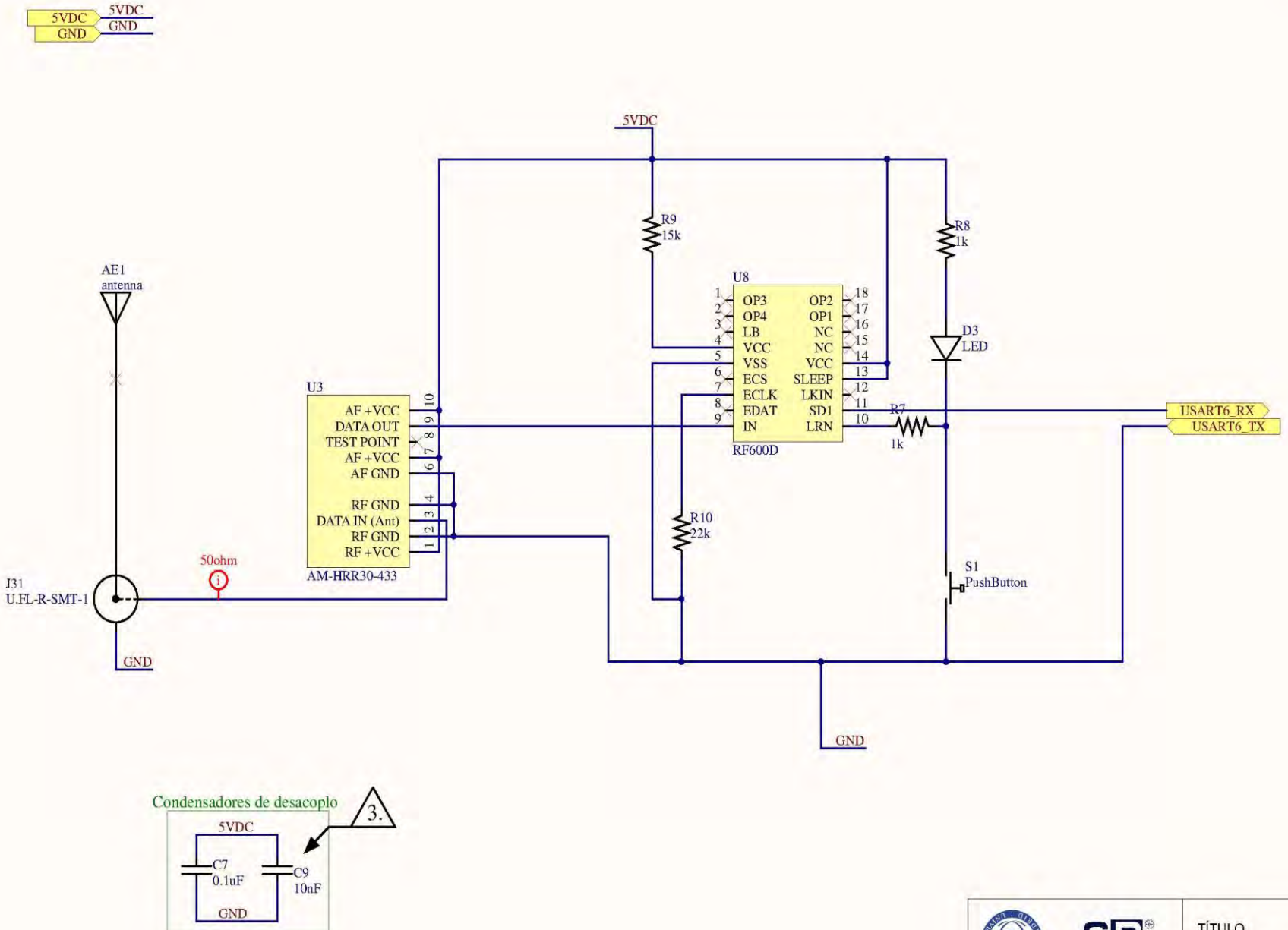
Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Sistema de Audio

ESCALA SCALE	FORMATO SIZE	HOJA SHEET
1/1	A4	7 / 8



DIBUJA	FECHA
Sergio Castillo	09/09/2016

Universidad Carlos III de Madrid

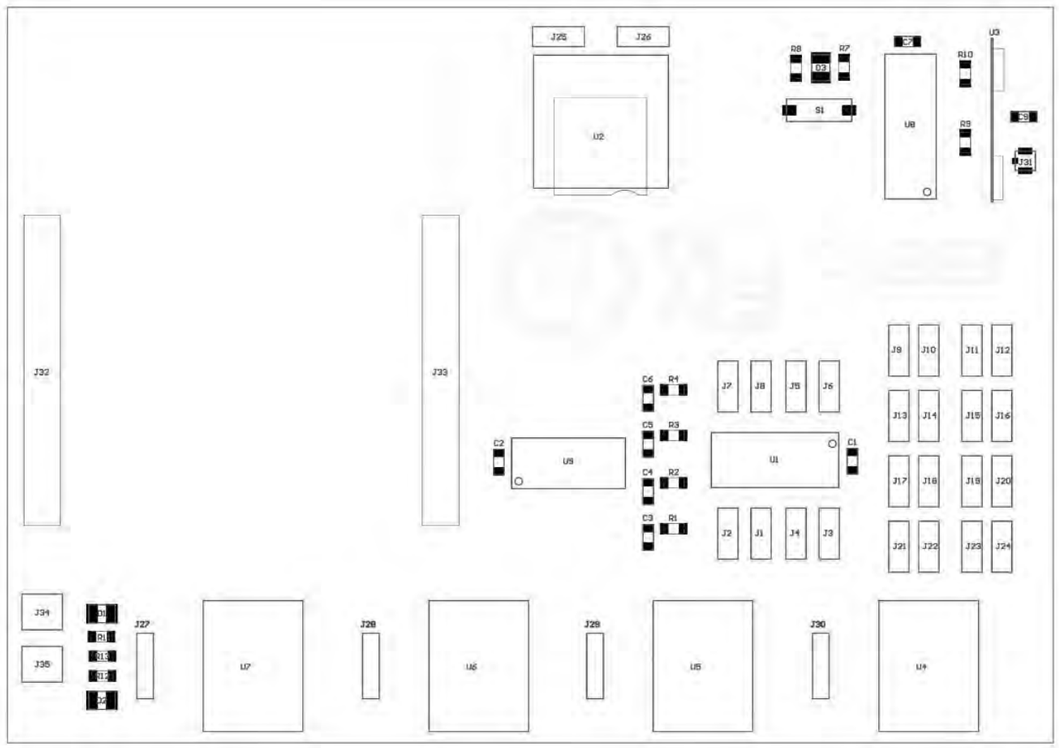
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Receptor Radiofrecuencia

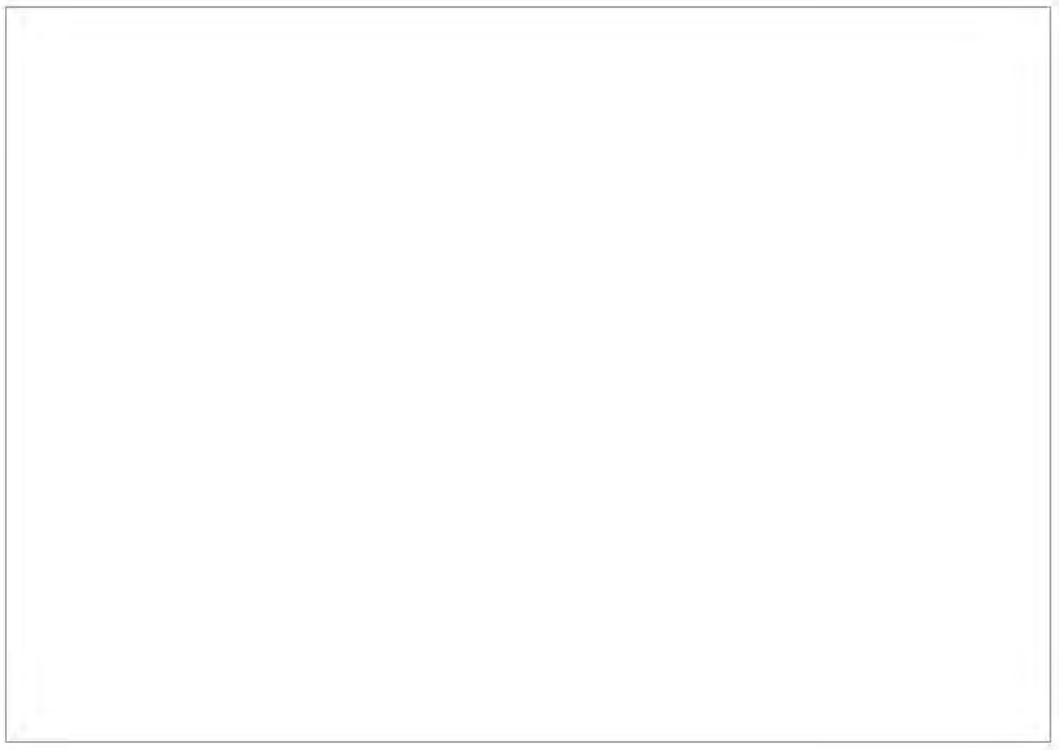
ESCALA SCALE	FORMATO SIZE	HOJA SHEET
1/1	A4	8 / 8

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid	PROYECTO: PCB Carriles	
	TITULO: Plano de Situacion de Componentes (Top View)	
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras.	ESCALA: 1 / 1	FORMATO: A3
Autores: Sergio Castillo Moheda no y Sergio Aguilera Sevilla		HOJA: 1/2

DIBUJA	FECHA
Sergio Castillo	09/09/2016



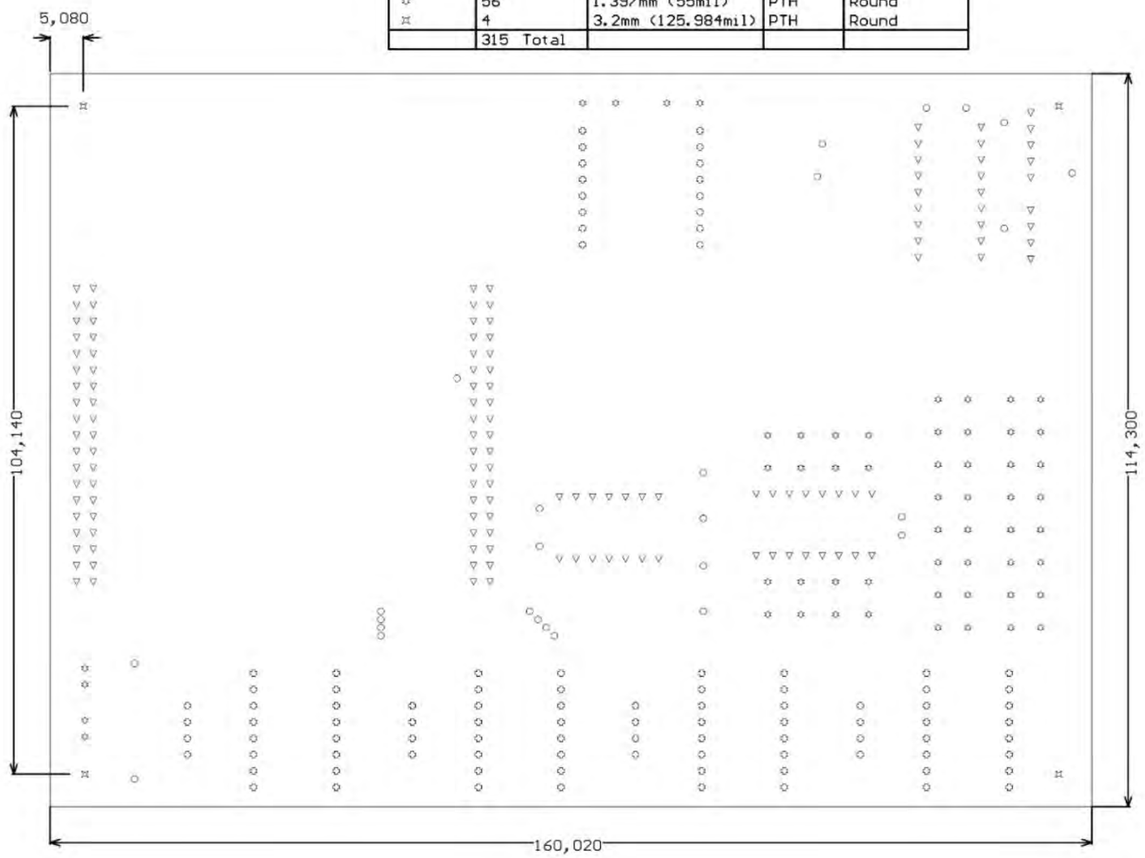
Universidad Complutense de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15). Transición a la Vida Adulta - Juego de Carreras.	TÍTULO Plano de Situacion de Componentes (Bottom View)			
	ESCALA 1 / 1	FORMATO A3	HOJA 2/2	

DIBUJA	FECHA
Sergio Castillo	09/09/2016

TITLE	SHEETS	TYPE	LAYER	FILE
Master Drawing Side	2/11	Pdf	1	PCB_Carriles.GTL
" " "	3/11	"	2	PCB_Carriles.G1
" " "	4/11	"	3	PCB_Carriles.G2
" " "	5/11	"	4	PCB_Carriles.GBL
Master Drawing Solder	6/11	"	On Layer 1	PCB_Carriles.GTS
" " "	7/11	"	On Layer 4	PCB_Carriles.GBS
Master Drawing Paste	8/11	"	On Layer 1	PCB_Carriles.GTP
Master Drawing Paste	9/11	"	On Layer 4	PCB_Carriles.GBP
Master Drawing Silkscreen	10/11	"	On Layer 1	PCB_Carriles.GTO
Master Drawing Silkscreen	11/11	"	On Layer 4	PCB_Carriles.GBO

LAYERSTACK

Symbol	Hit Count	Tool Size	Plated	Hole Type
○	26	0.3mm (11.811mil)	PTH	Round
▽	133	0.762mm (30mil)	PTH	Round
○	96	1.016mm (40mil)	PTH	Round
☼	56	1.397mm (55mil)	PTH	Round
⌗	4	3.2mm (125.984mil)	PTH	Round
	315 Total			



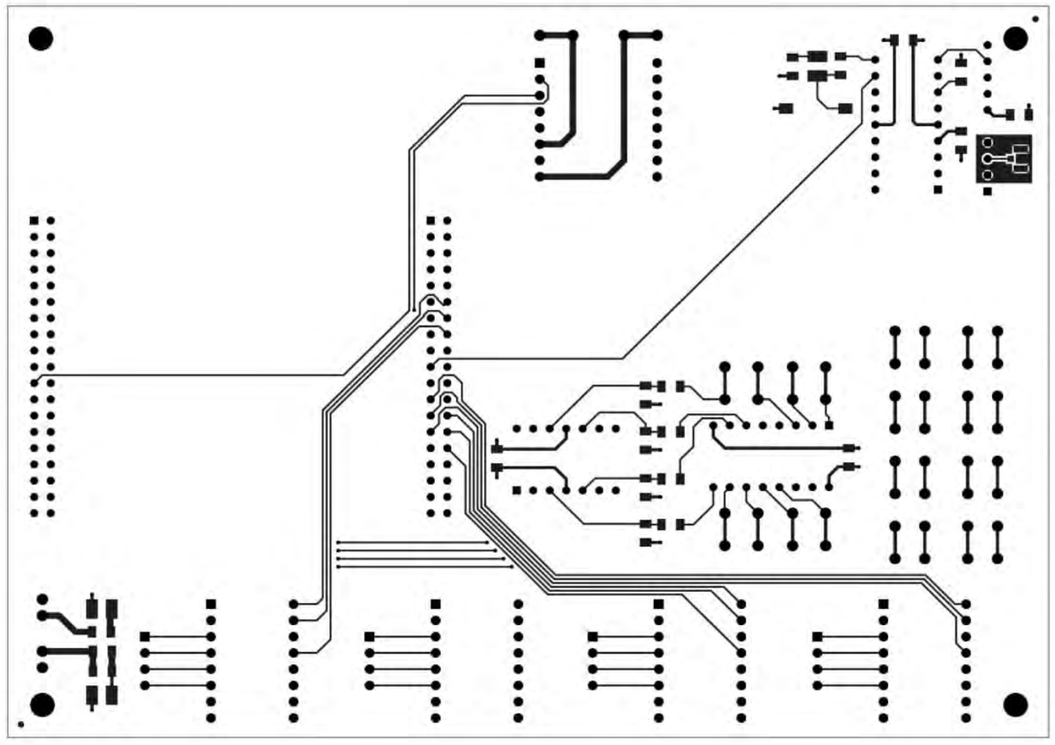
Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras.

Autores: Sergio Castillo Mohedano y Sergio Aguilera Sevilla

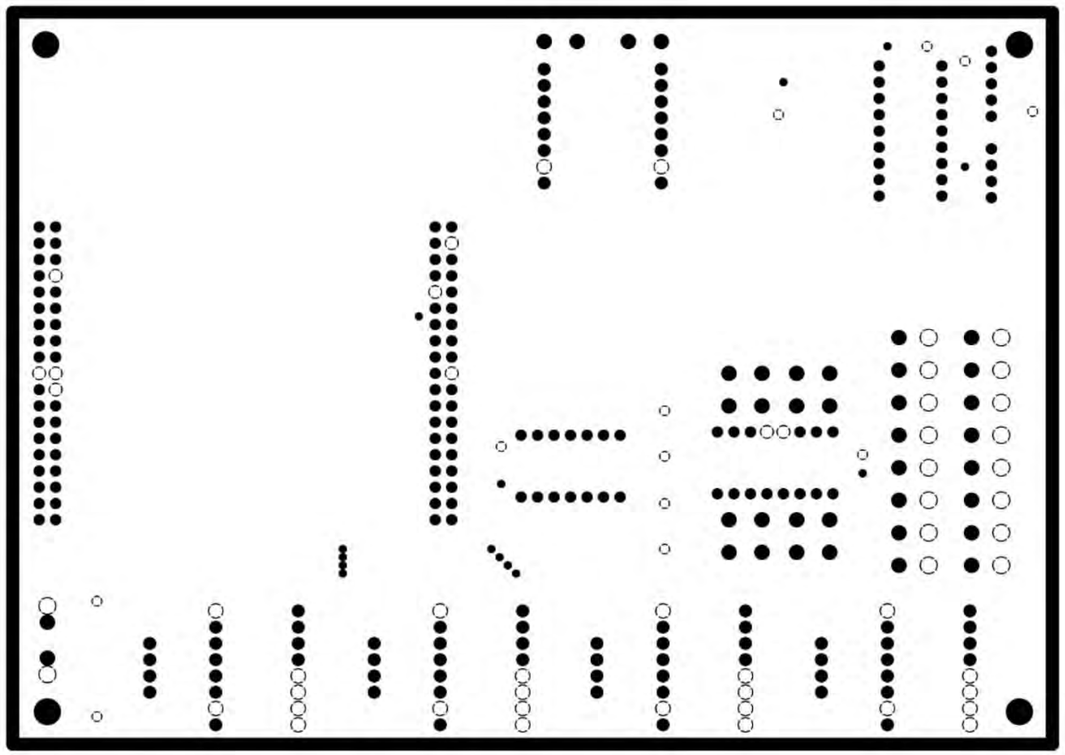
PROYECTO	
PCB Carriles	
TITULO	
Drill Guide For (Bottom Layer,Top Layer)	
ESCALA 1 / 1	FORMATO A3
HOJA 1/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



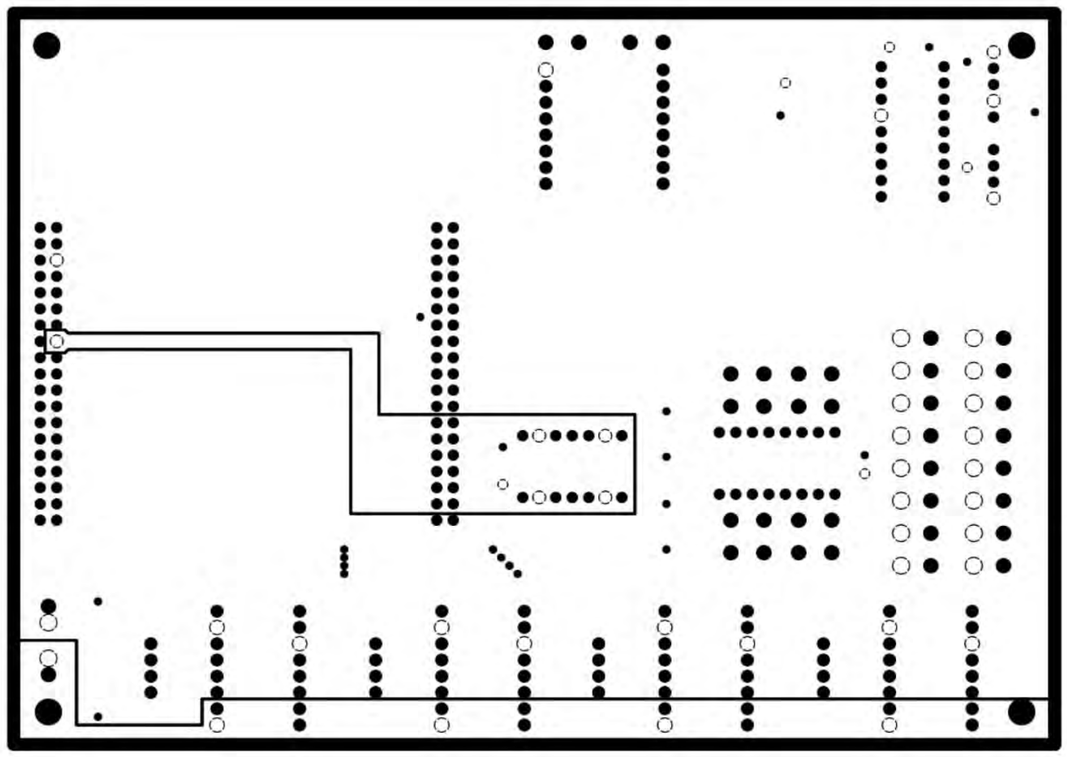
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15) Transición a la Vida Adulta - Juego de Carreras	TITULO Top Layer			
	ESCALA 1 / 1	FORMATO A3	HOJA 2/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para asimnce del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras	TITULO Internal Plane 1			
	ESCALA 1 / 1	FORMATO A3	HOJA 3/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016

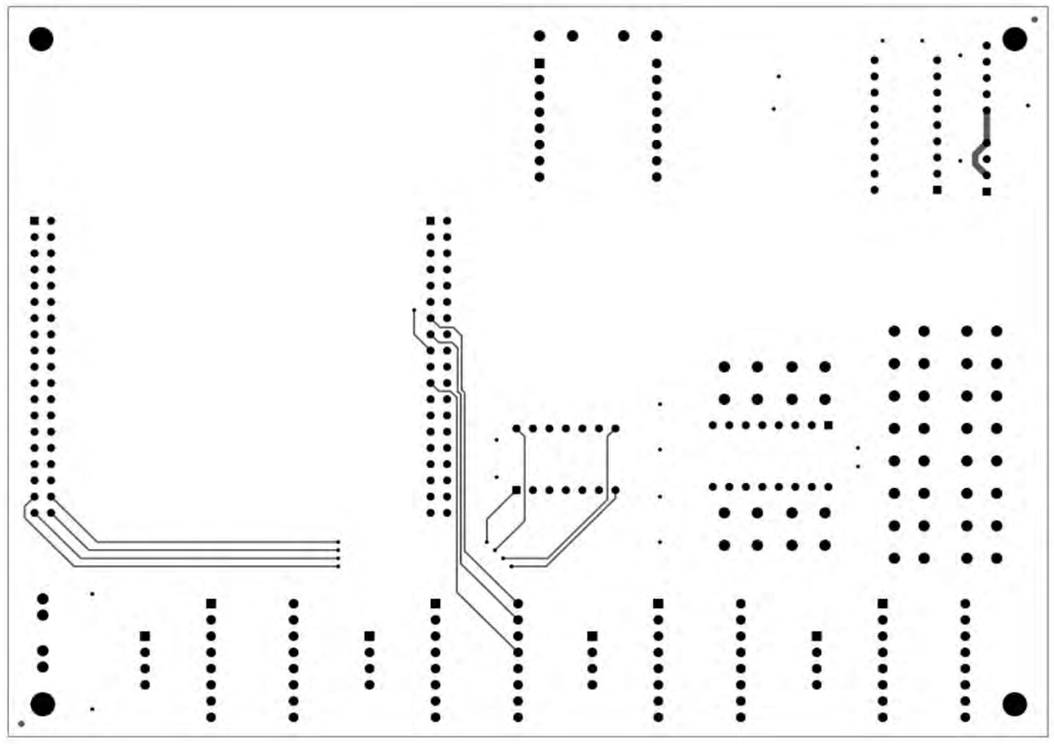


Universidad Carlos III de Madrid
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15)
Transición a la Vida Adulta - Juego de Carreras

TITULO: Internal Plane 2

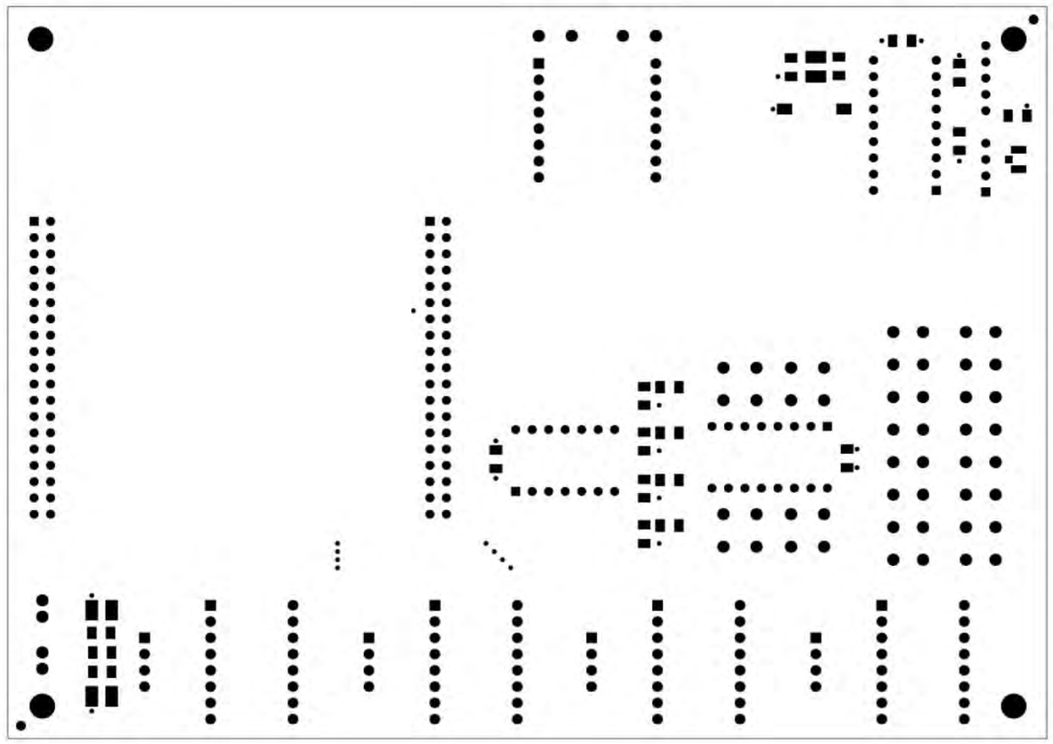
ESCALA	FORMATO	HOJA
1 / 1	A3	4/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016



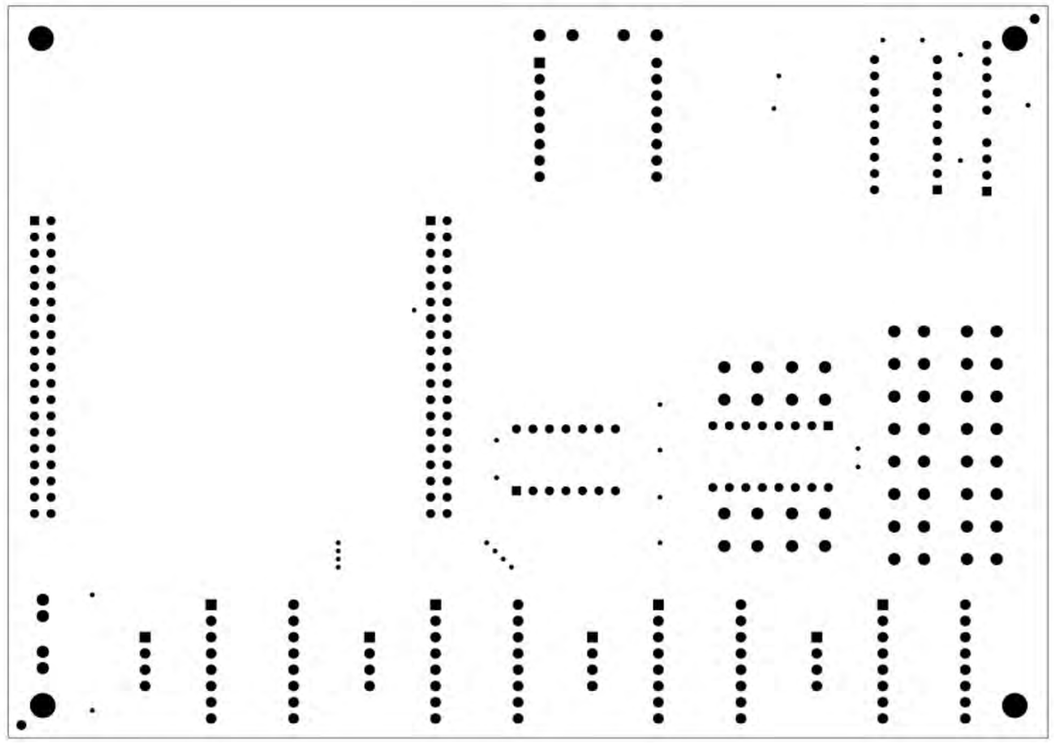
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras	TITULO Bottom Layer			
	ESCALA 1 / 1	FORMATO A3	HOJA 5/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



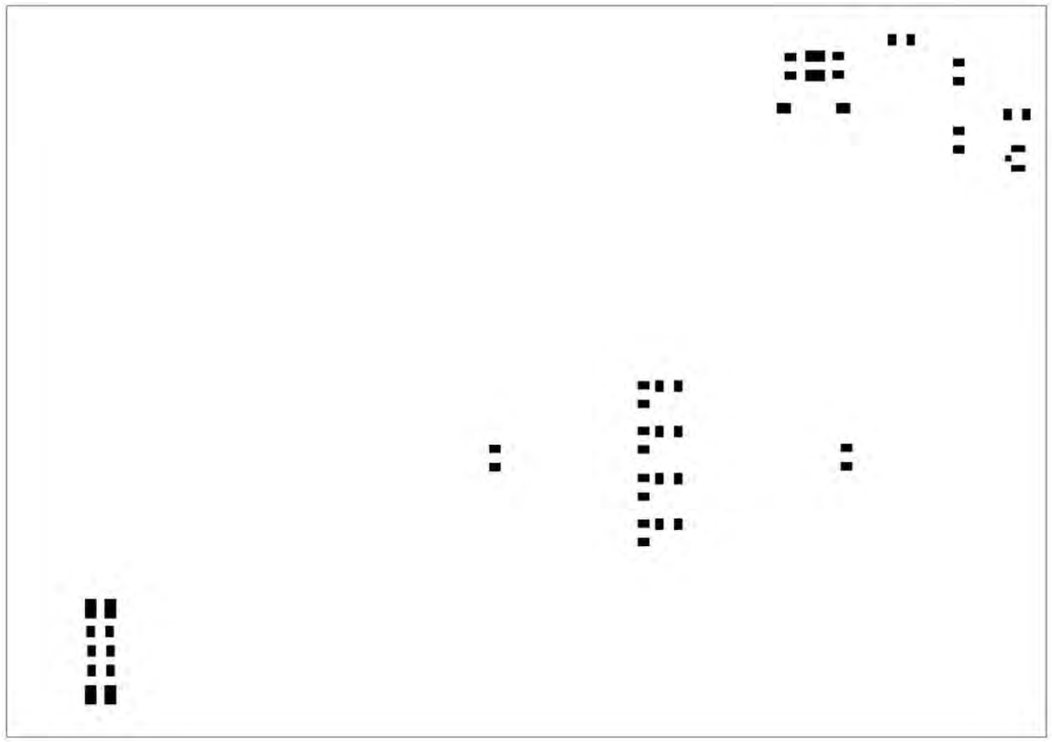
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15) Transición a la Vida Adulta - Juego de Carreras	ESCALA 1 / 1	FORMATO A3	HOJA 6/11	TITULO Top Solder Mask Print

DIBUJA	FECHA
Sergio Castillo	09/09/2016



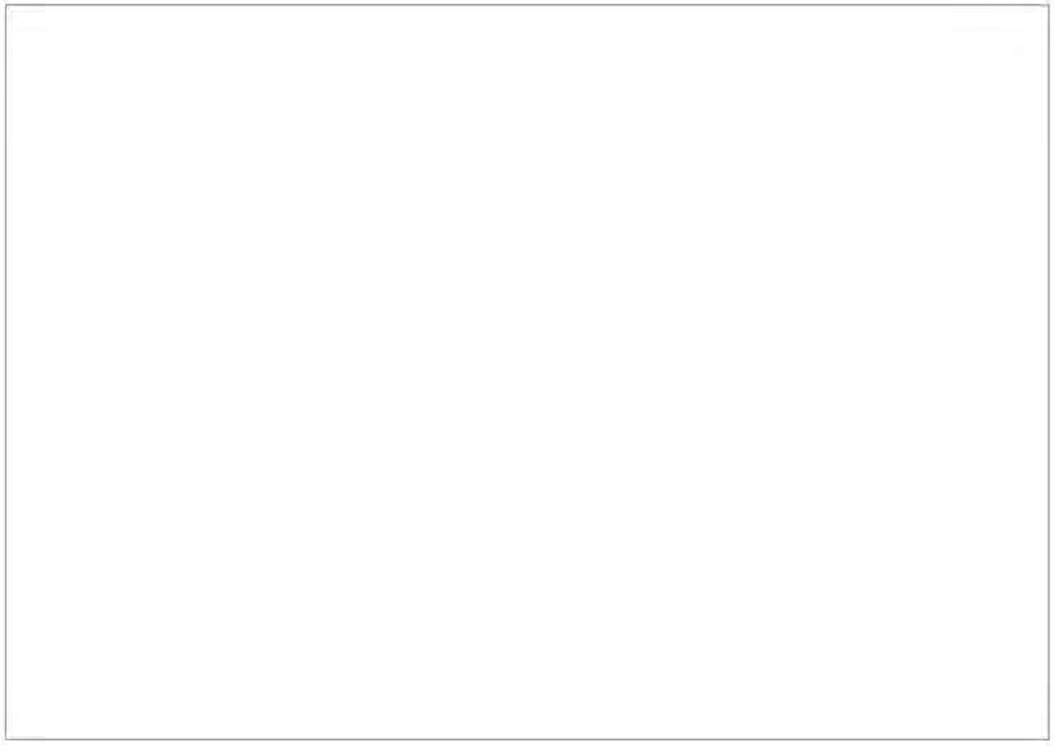
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras	TITULO Bottom Solder Mask Print		
	ESCALA 1 / 1	FORMATO A3	HOJA 7/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016



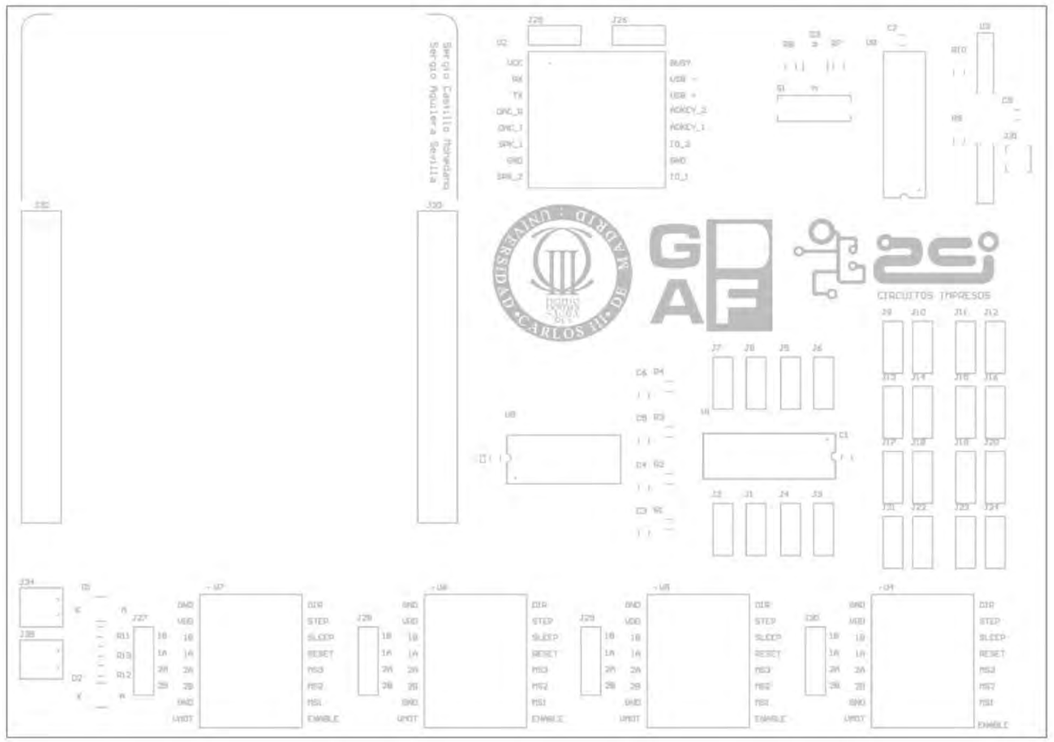
Universidad Carlos III de Madrid Desarrollo de una sonda térmica para alumnos del Colegio San Rafael (15) Transición a la Vida Adulta - Colegio de Carmelitas	TÍTULO Top Paste Mask Print			
	ESCALA 1 / 1	FORMATO A3	HOJA 8/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



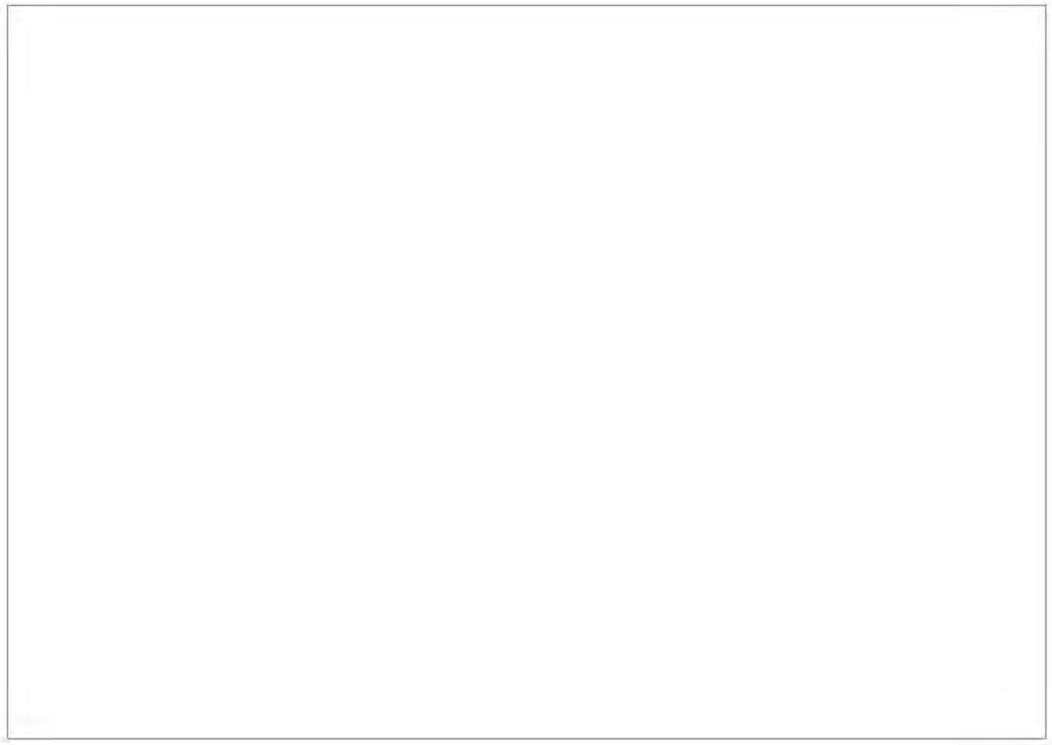
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15) Translación a la Vida Adulta - Juego de Carreras.	TITULO: Bottom Paste Mask Print ESCALA: 1 / 1 · FORMATO: A3 · HOJA: 9/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras.	TITULO Top Silkscreen Overlay		
	ESCALA 1 / 1	FORMATO A3	HOJA 10/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15)
Translación a la Vida Activa - Juego de Carrera

TITULO: Bottom Silkscreen Overlay

ESCALA	FORMATO	HOJA
1 / 1	A3	11/11

DIBUJA	FECHA
Sergio Castillo	10/09/2016

NOTAS:

Aquellos componentes en azul no pertenecen a la placa, si no a un nivel superior. Se muestran en los esquemas con tal de facilitar la comprensión de los circuitos.

1. Accionamiento por flanco de subida, por tanto se colocan resistencias R17, R19 y R21 hacia GND y se sueldan puentes (Solder Bridges) SB8, SB7 y SB6.

2. Accionamiento por flanco de subida, por tanto se sueldan puentes (Solder Bridges) SB10 y SB2*.

3.* Cambio realizado a posteriori sobre la placa. Se desolda SB2 y se coloca en paralelo una resistencia de 1kohm.

SISTEMA CARRILES

HOJA	DESCRIPCIÓN
1	ÍNDICE
2	GLOBAL
3	MICROCONTROLADOR
4	ALIMENTACIÓN
5	SERVOMOTORES
6	PANEL DE CONTROL
7	INFRARROJOS
8	RADIOFRECUENCIA
9	CIRCUITO ACCIONAMIENTO

Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

Autores
Sergio Castillo Mohedano
Sergio Aguilera Sevilla

PROYECTO
PCB Rampa

TÍTULO
Índice

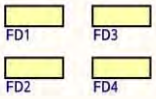
ESCALA SCALE	FORMATO SIZE	HOJA SHEET
1/1	A4	1 / 9

DIBUJA	FECHA
Sergio Castillo	10/09/2016

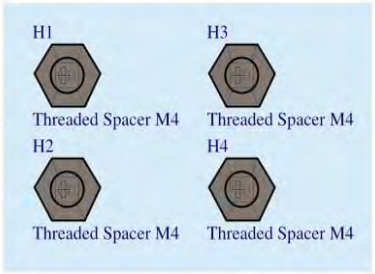
Logo GDAF

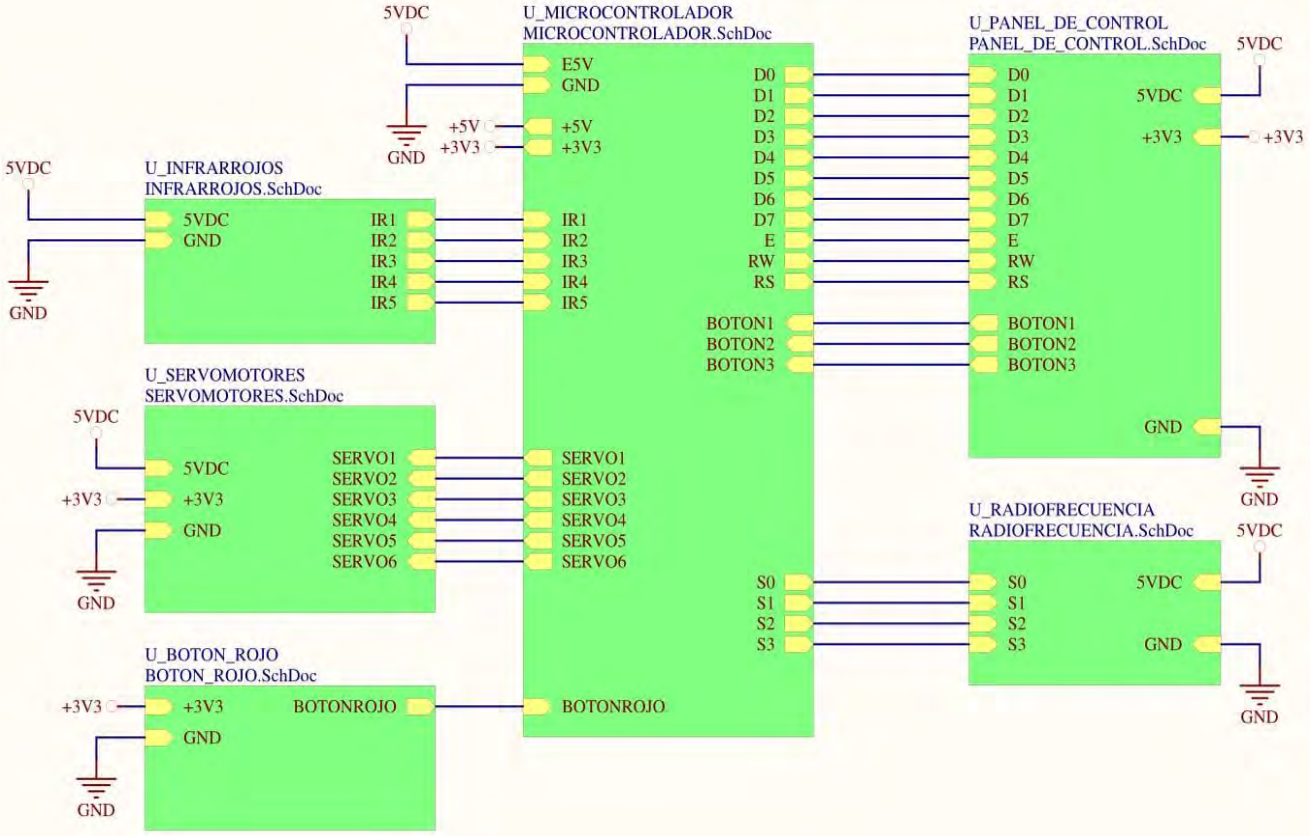

Logo UC3M

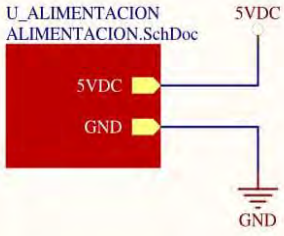

Logo 2CISA



MH1
DRILL 4 mm PTH
MH2
DRILL 4 mm PTH
MH3
DRILL 4 mm PTH
MH4
DRILL 4 mm PTH







Universidad Carlos III de Madrid

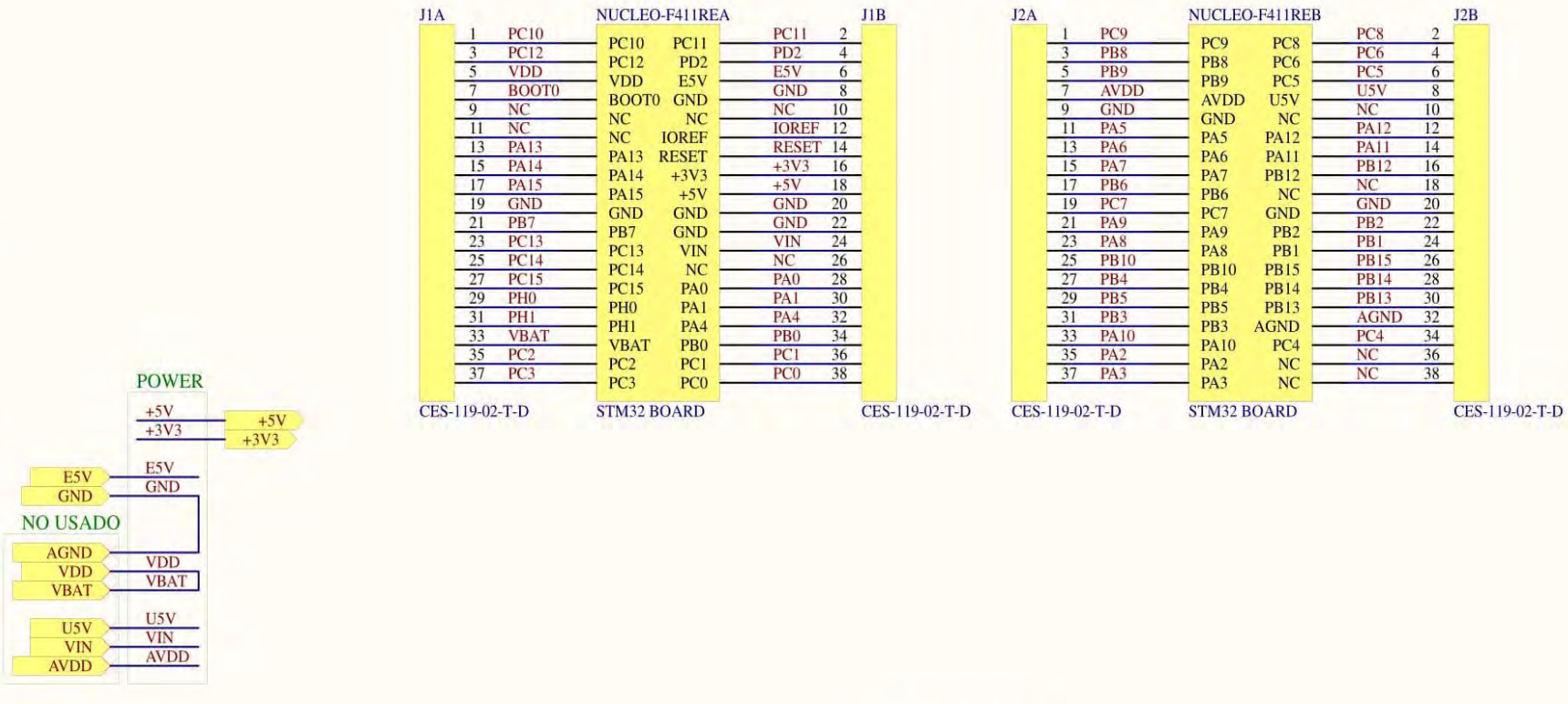
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Esquema Global

ESCALA / SCALE	FORMATO / SIZE	HOJA / SHEET
1/1	A4	2 / 9

DIBUJA	FECHA
Sergio Castillo	10/09/2016



Universidad Carlos III de Madrid

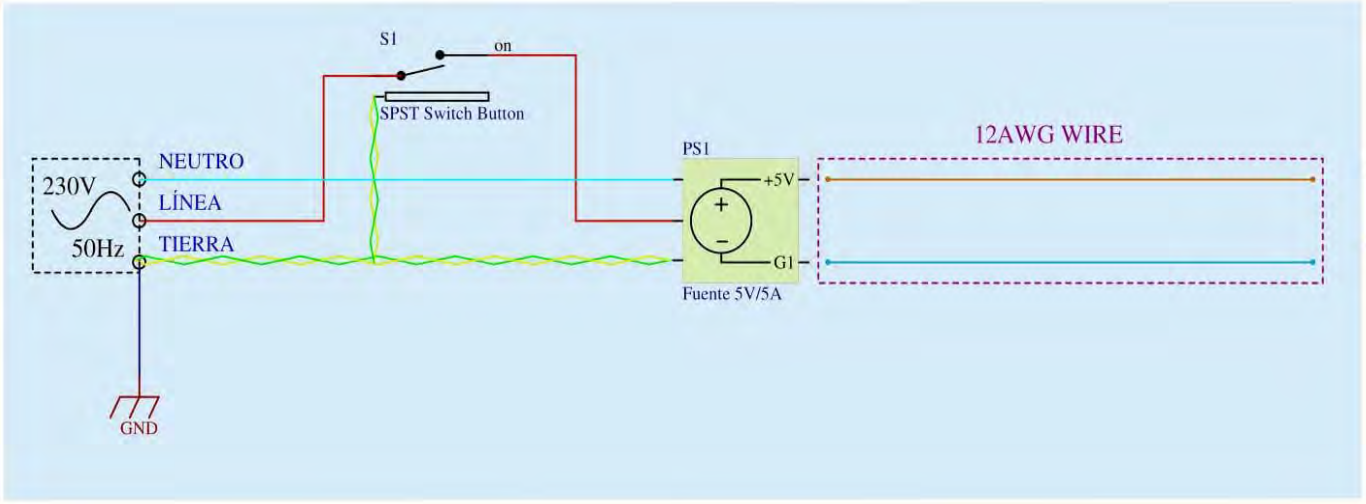
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

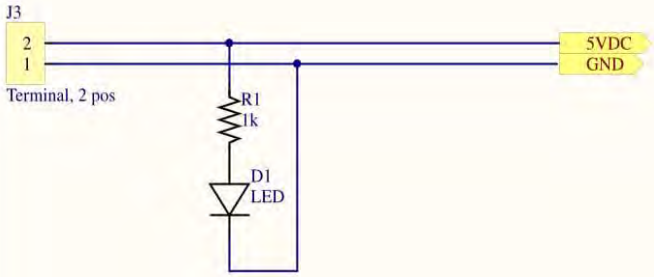
TÍTULO

Microcontrolador

ESCALA / SCALE	FORMATO / SIZE	HOJA / SHEET
1/1	A4	3 / 9

DIBUJA	FECHA
Sergio Castillo	10/09/2016





Universidad Carlos III de Madrid

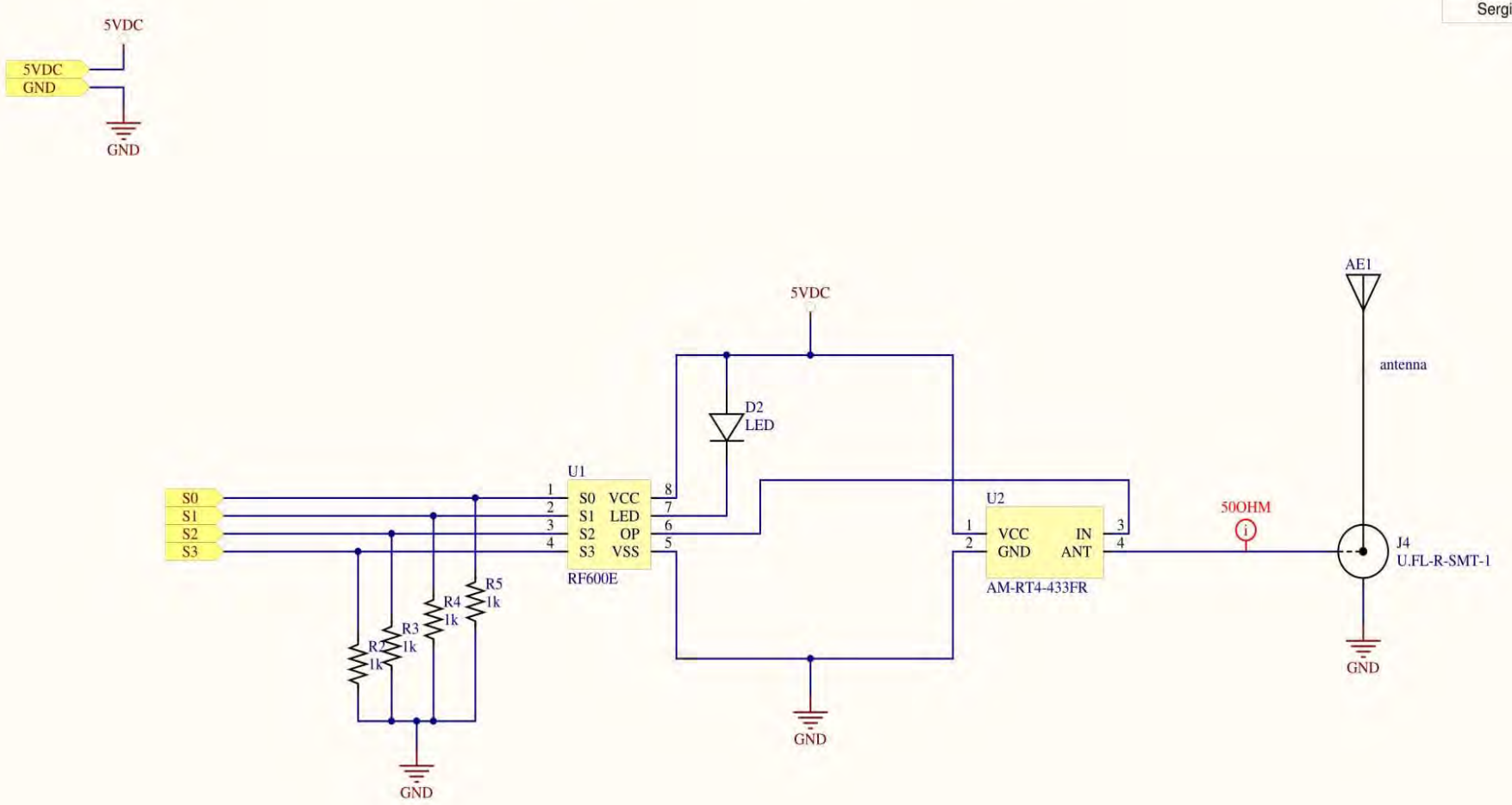
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

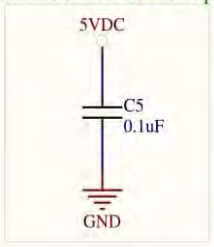
Alimentación

ESCALA SCALE	FORMATO SIZE	HOJA SHEET
1/1	A4	4 / 9

DIBUJA	FECHA
Sergio Castillo	10/09/2016



Condensadores de desacoplo



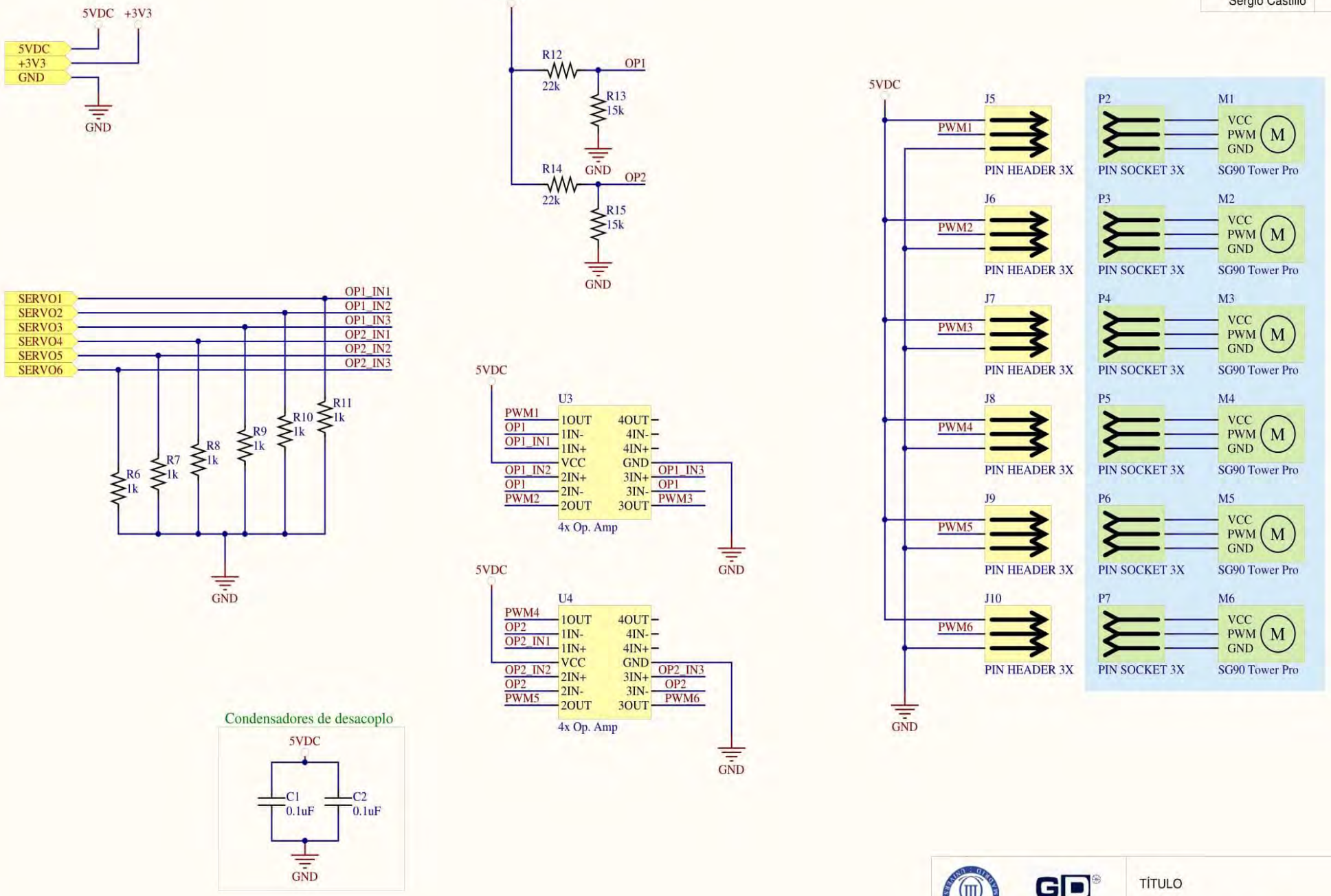
Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

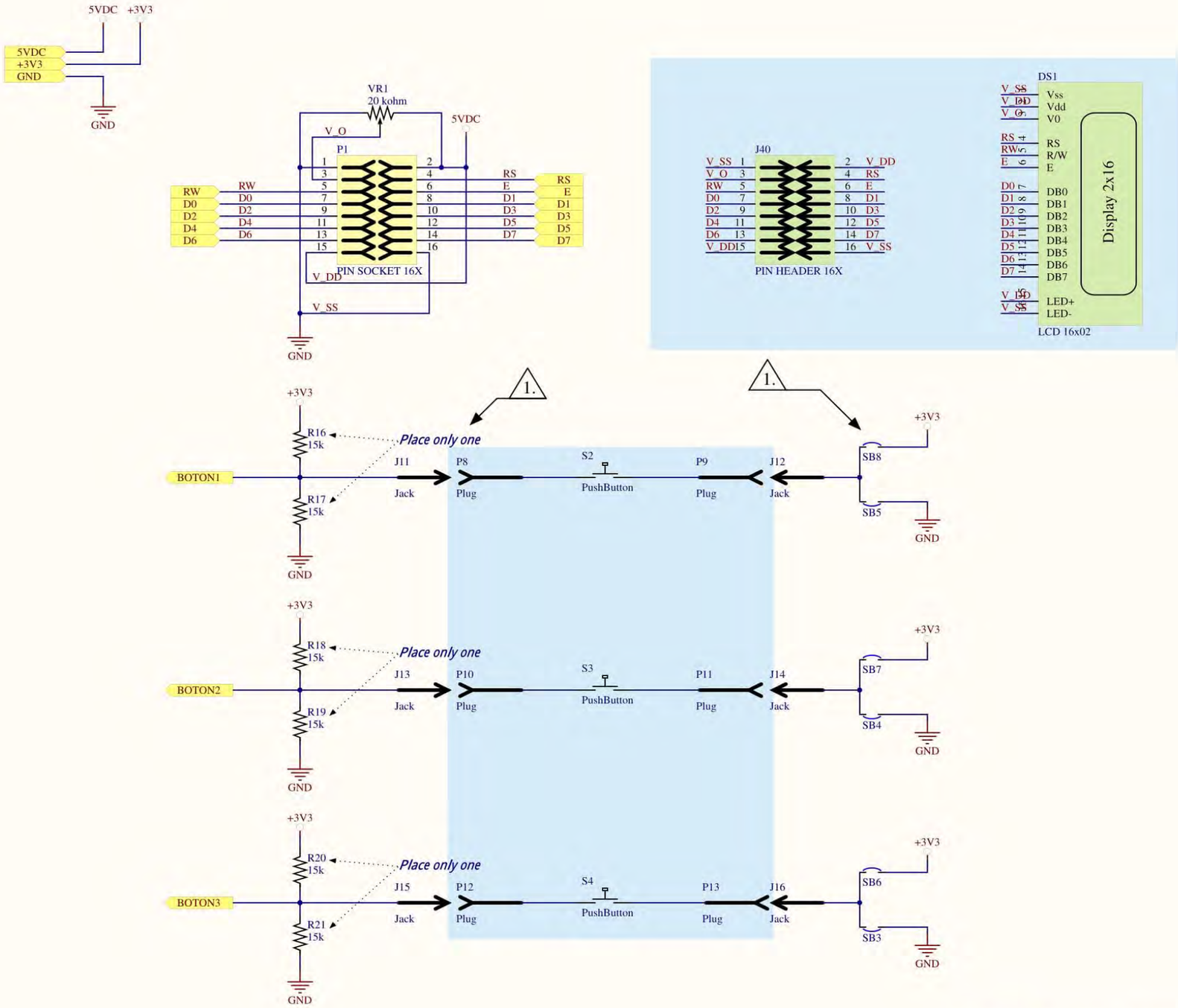
TÍTULO

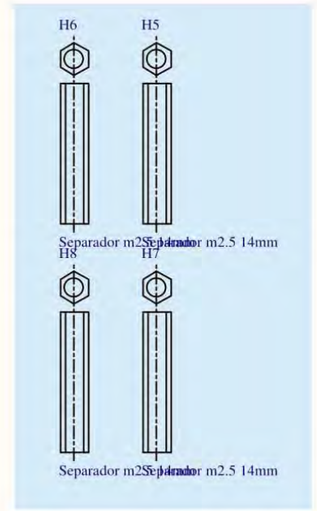
Transmisor Radiofrecuencia

ESCALA SCALE	FORMATO SIZE	HOJA SHEET
1/1	A4	5 / 9


DIBUJA
Sergio Castillo
FECHA
10/09/2016
5VDC
+3V3
GND
R12
22k
R13
15k
R14
22k
R15
15k
OP1
OP2
SERVO1
SERVO2
SERVO3
SERVO4
SERVO5
SERVO6
OP1_IN1
OP1_IN2
OP1_IN3
OP2_IN1
OP2_IN2
OP2_IN3
R6
R7
R8
R9
R10
R11
1k
U3
U4
PWM1
PWM2
PWM3
PWM4
PWM5
PWM6
1OUT
1IN-
1IN+
VCC
2IN+
2IN-
2OUT
4OUT
4IN-
4IN+
GND
3IN+
3IN-
3OUT
4x Op. Amp
Condensadores de desacoplo
C1
0.1uF
C2
0.1uF
J5
J6
J7
J8
J9
J10
PIN HEADER 3X
P2
P3
P4
P5
P6
P7
PIN SOCKET 3X
M1
M2
M3
M4
M5
M6
VCC
PWM
GND
M
SG90 Tower Pro
Universidad Carlos III de Madrid
Desarrollo de una ayuda técnica
para alumnos del Colegio San Rafael (15):
Transición a la vida Adulta - Juego de Carreras.
TÍTULO
Servomotores
ESCALA
SCALE
1/1
FORMATO
SIZE
A4
HOJA
SHEET
6 / 9

DIBUJA	FECHA
Sergio Castillo	10/09/2016





Universidad Carlos III de Madrid

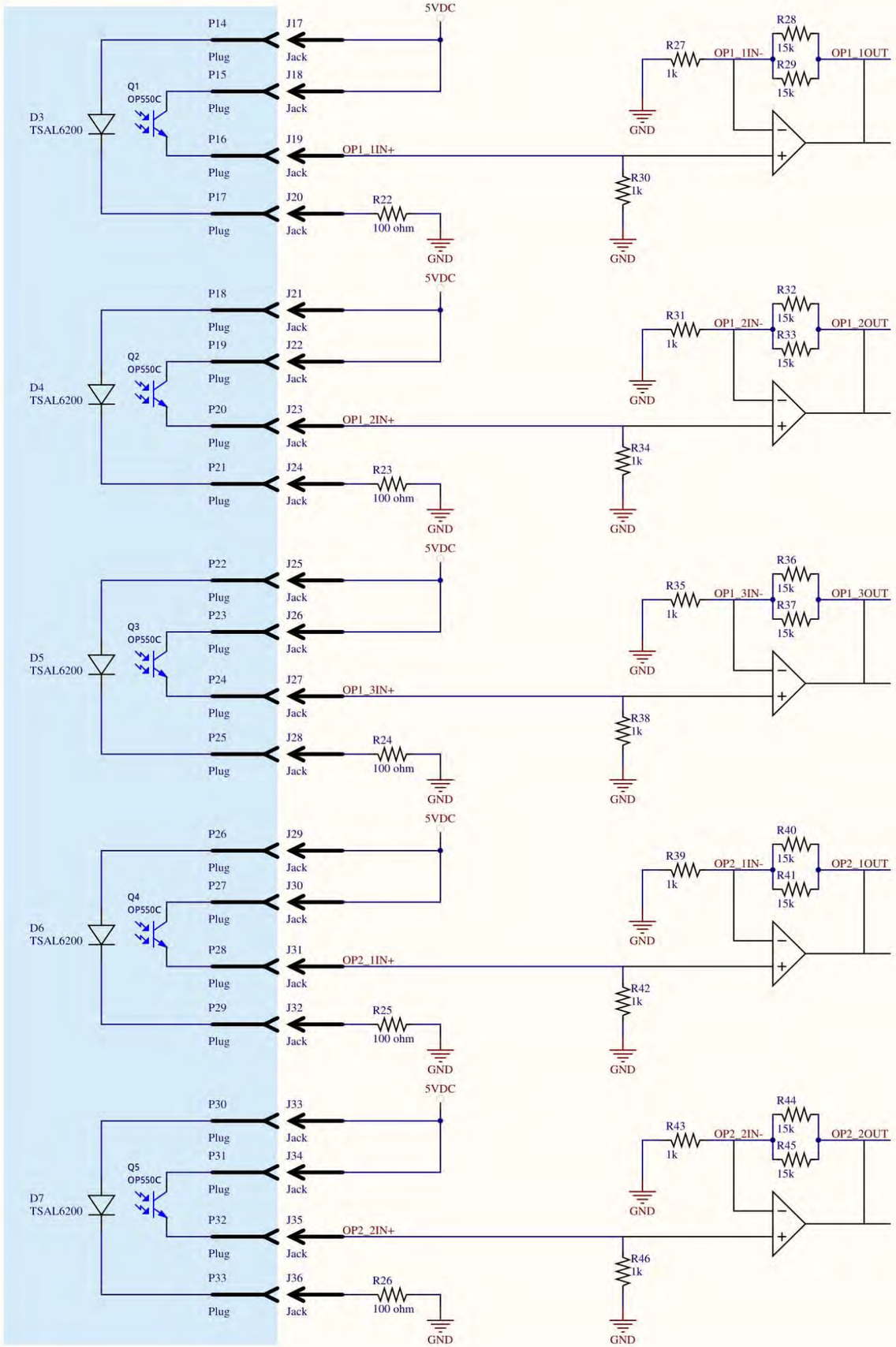
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

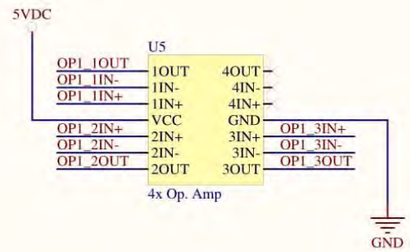
TÍTULO

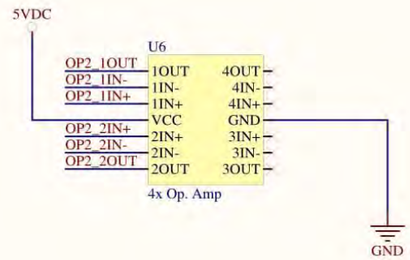
Panel de Control

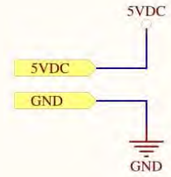
ESCALA SCALE	1/1	FORMATO SIZE	A3	HOJA SHEET	7 / 9

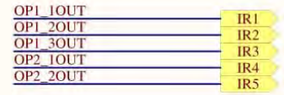
DIBUJA	FECHA
Sergio Castillo	10/09/2016











Condensadores de desacoplo

Universidad Carlos III de Madrid

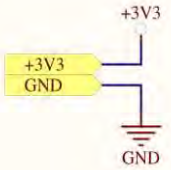
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

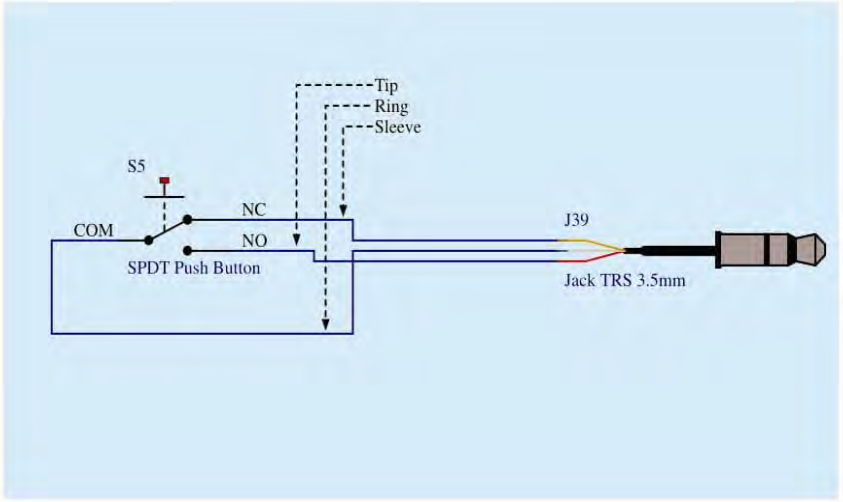
TÍTULO

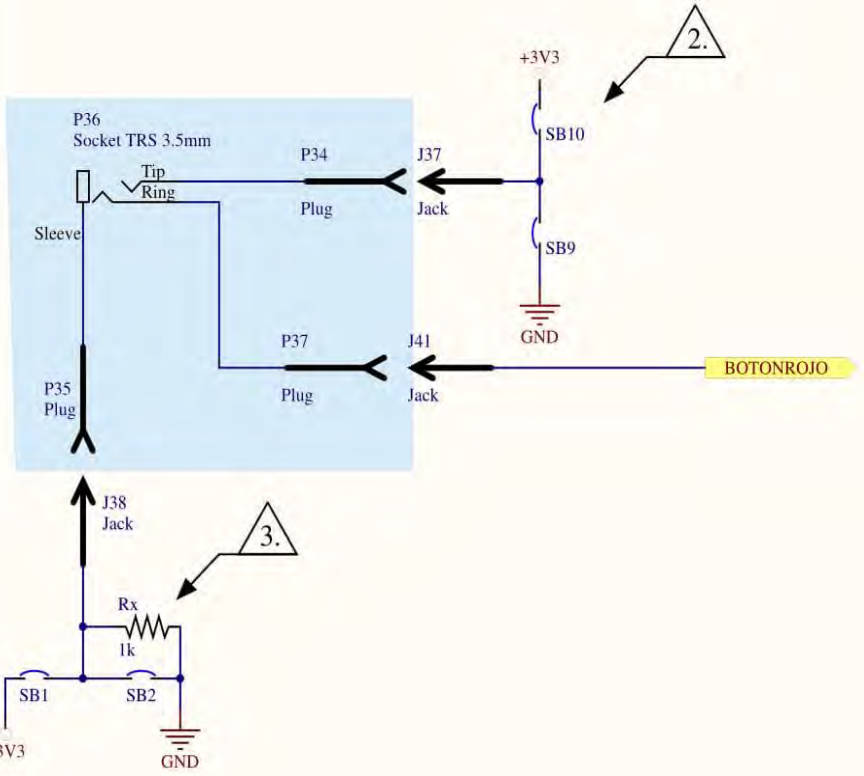
Infrarrojos

ESCALA SCALE	1/1	FORMATO SIZE	A3	HOJA SHEET	8 / 9

DIBUJA	FECHA
Sergio Castillo	10/09/2016







Universidad Carlos III de Madrid

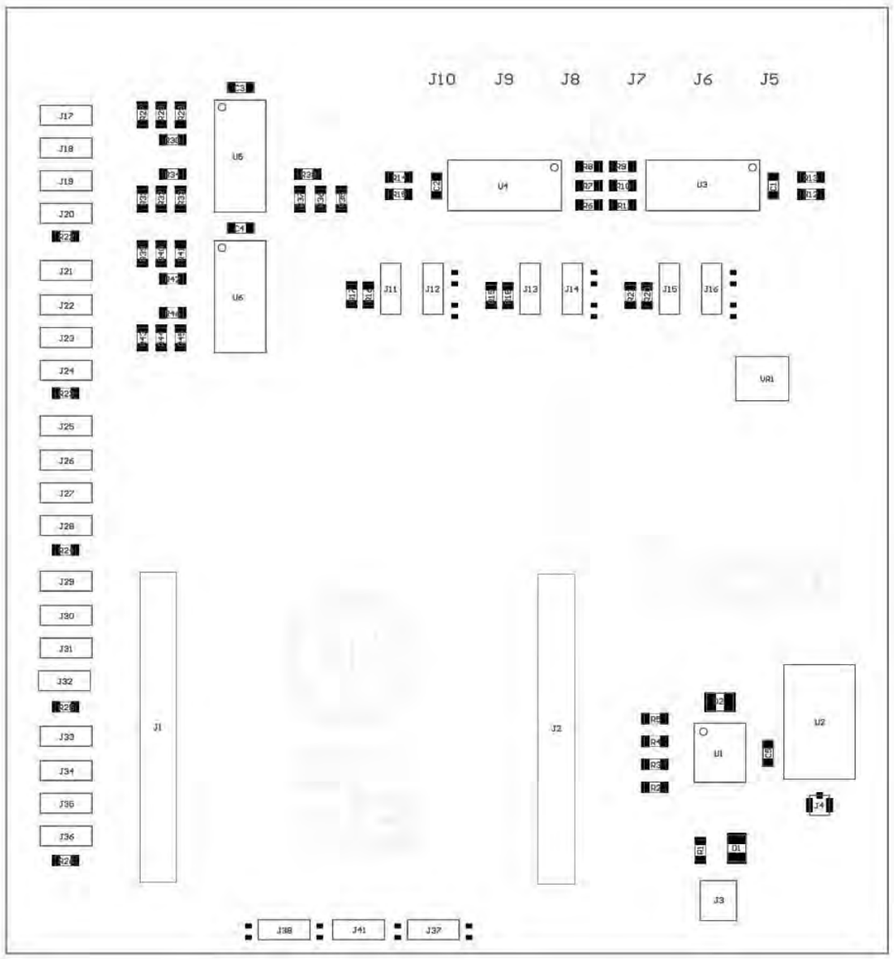
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la vida Adulta - Juego de Carreras.

TÍTULO

Circuito de Accionamiento

ESCALA SCALE	FORMATO SIZE	HOJA SHEET
1/1	A4	9 / 9

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras.

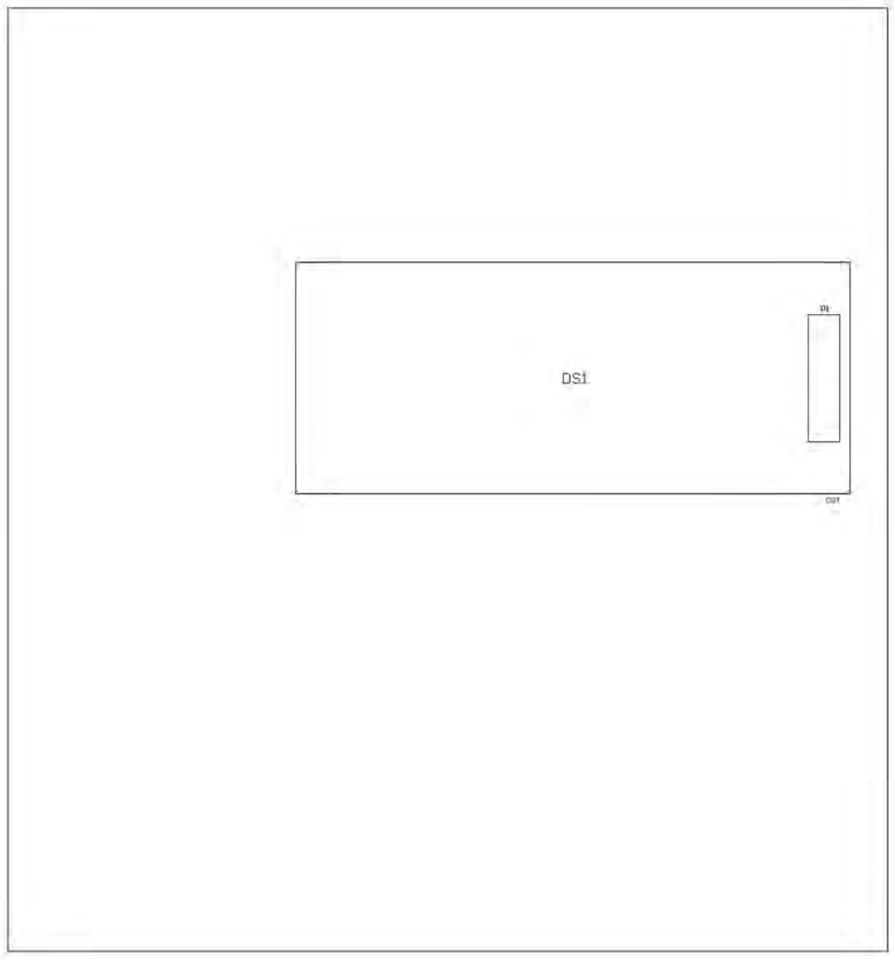
Autores:
Sergio Castillo Mohedano y Sergio Aguilera Sevilla

PROYECTO
PCB Rampa

TITULO
Plano de Situacion de Componentes (Top View)

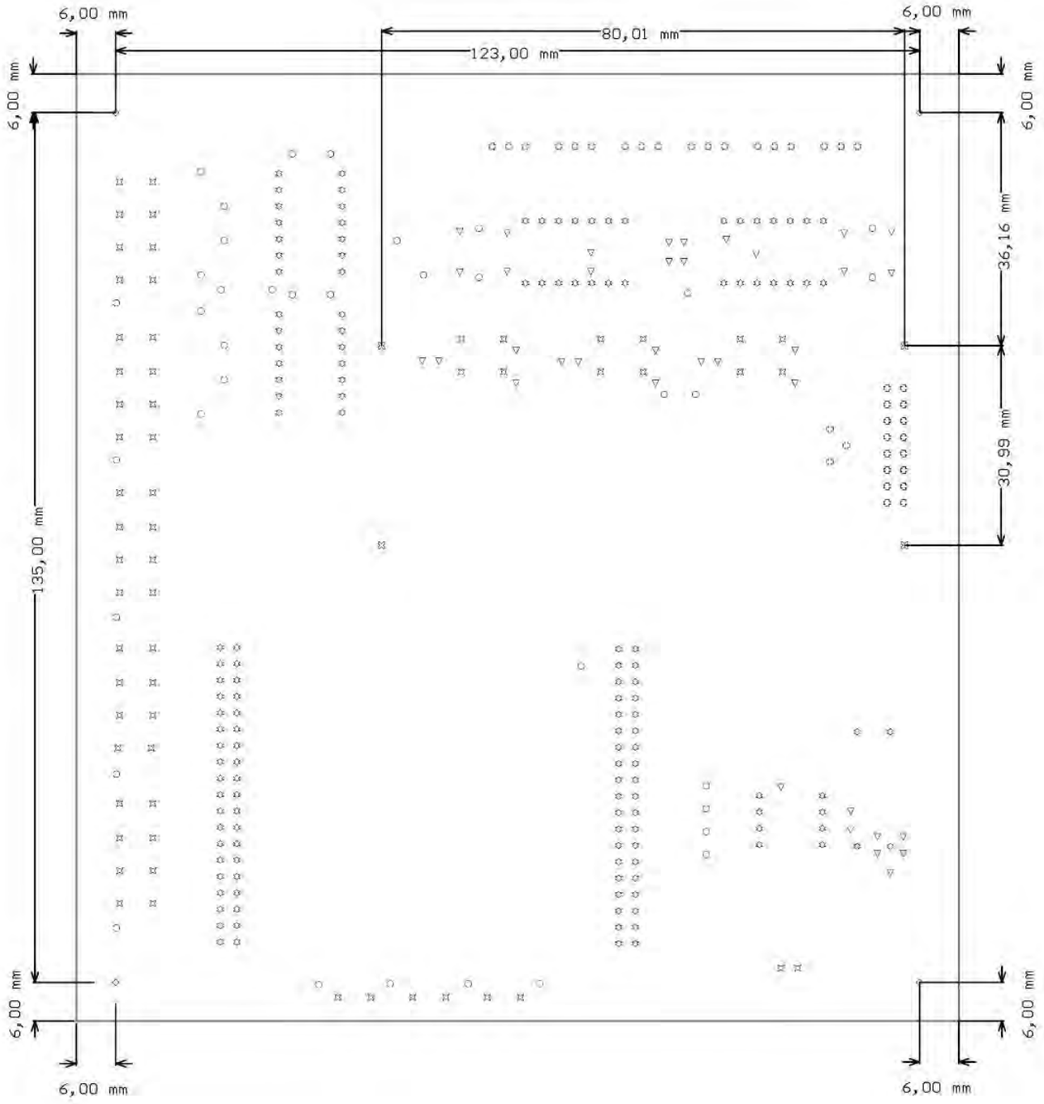
ESCALA	FORMATO	HOJA
1 / 1	A3	1/2

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15). Transición a la Vida Adulta - Juego de Carreras.	TITULO Plano de Situacion de Componentes (Bottom View)		
	ESCALA 1 / 1	FORMATO A3	HOJA 2/2

DIBUJA	FECHA
Sergio Castillo	09/09/2016



LAYERSTACK

TITLE	SHEETS	TYPE	LAYER	FILE
Master Drawing Side	2/11	Pdf	1	PCB_Rampa.GTL
" " "	3/11	"	2	PCB_Rampa.G1
" " "	4/11	"	3	PCB_Rampa.G2
" " "	5/11	"	4	PCB_Rampa.GBL
Master Drawing Solder	6/11	"	On Layer 1	PCB_Rampa.GTS
" " "	7/11	"	On Layer 4	PCB_Rampa.GBS
Master Drawing Paste	8/11	"	On Layer 1	PCB_Rampa.GTP
Master Drawing Paste	9/11	"	On Layer 4	PCB_Rampa.GBP
Master Drawing Silkscreen	10/11	"	On Layer 1	PCB_Rampa.GTO
Master Drawing Silkscreen	11/11	"	On Layer 4	PCB_Rampa.GBO

Symbol	Hit Count	Tool Size	Plated	Hole Type
▽	36	0.3mm (11.811mil)	PTH	Round
□	37	0.5mm (19.685mil)	PTH	Round
☼	144	0.762mm (30mil)	PTH	Round
☼	53	1.016mm (40mil)	PTH	Round
⌑	60	1.397mm (55mil)	PTH	Round
⊠	4	2.75mm (108.268mil)	PTH	Round
◇	4	4mm (157.48mil)	PTH	Round
	338 Total			

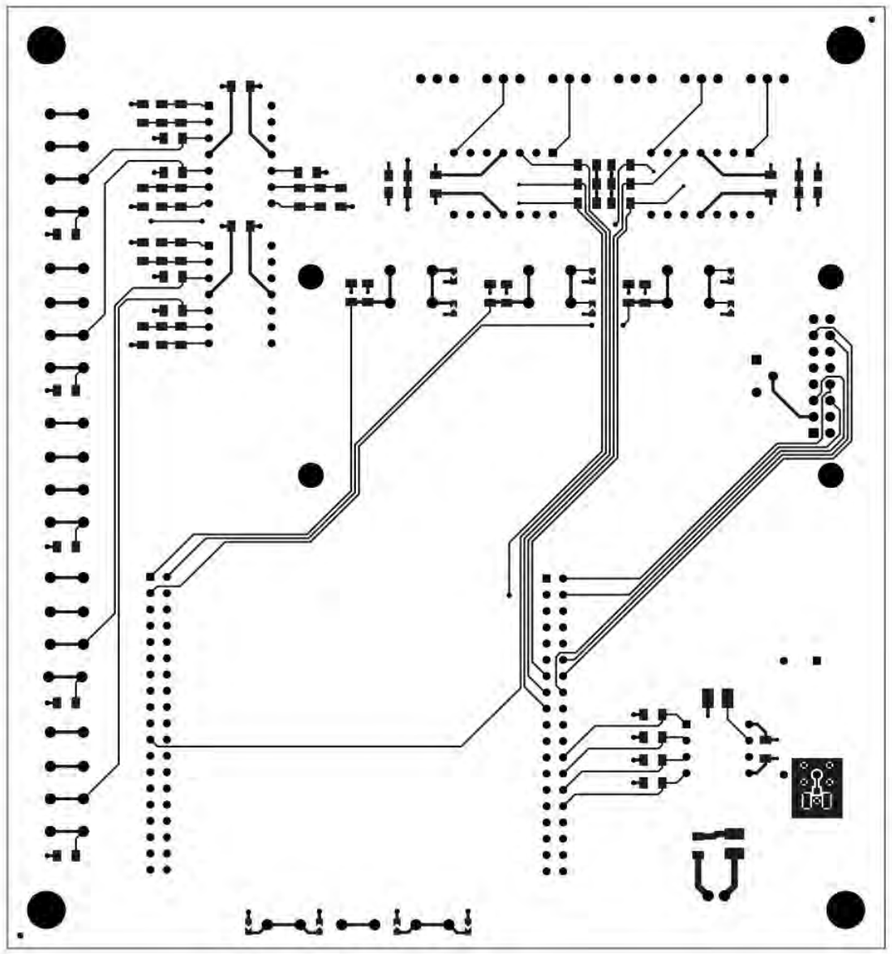
Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (Tfs): Transición a la Vida Adulta - Juego de Carreras

Autores:
Sergio Castillo Moheilano y Sergio Aguilera Sevilla

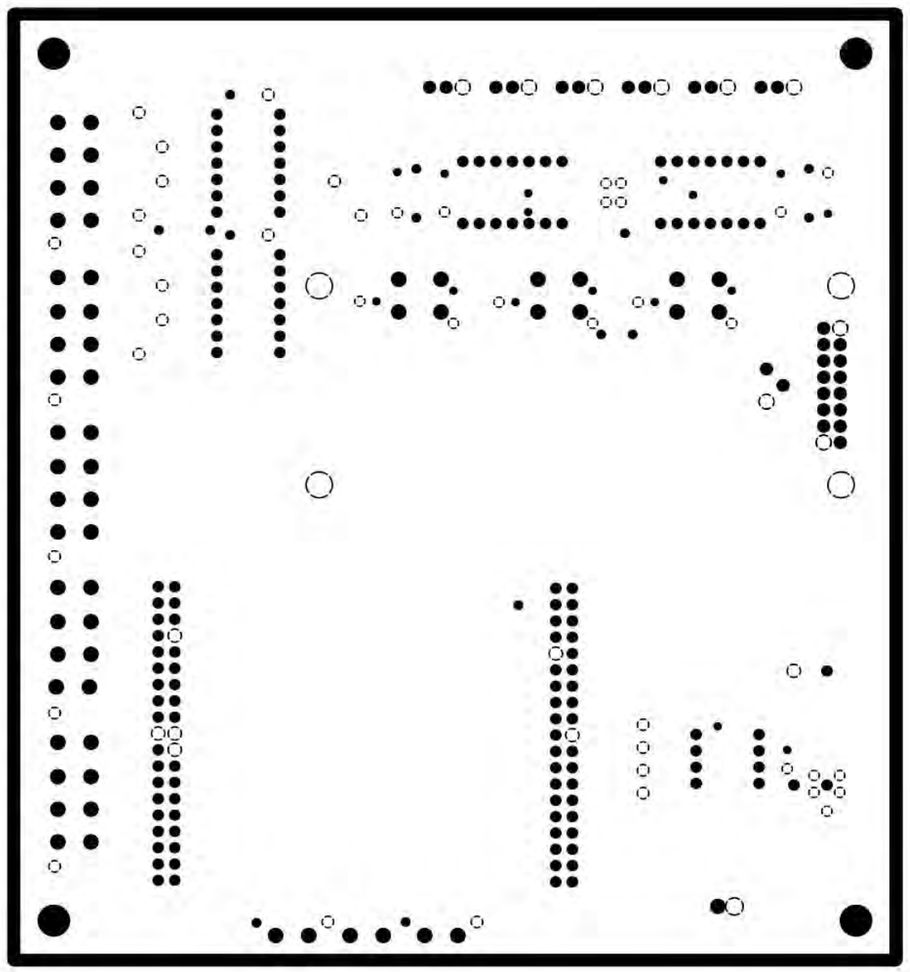
PROYECTO	
PCB Rampa	
TITULO	
Drill Guide For (Bottom Layer, Top Layer)	
ESCALA: 1 / 1	FORMATO: A3
HOJA: 1/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



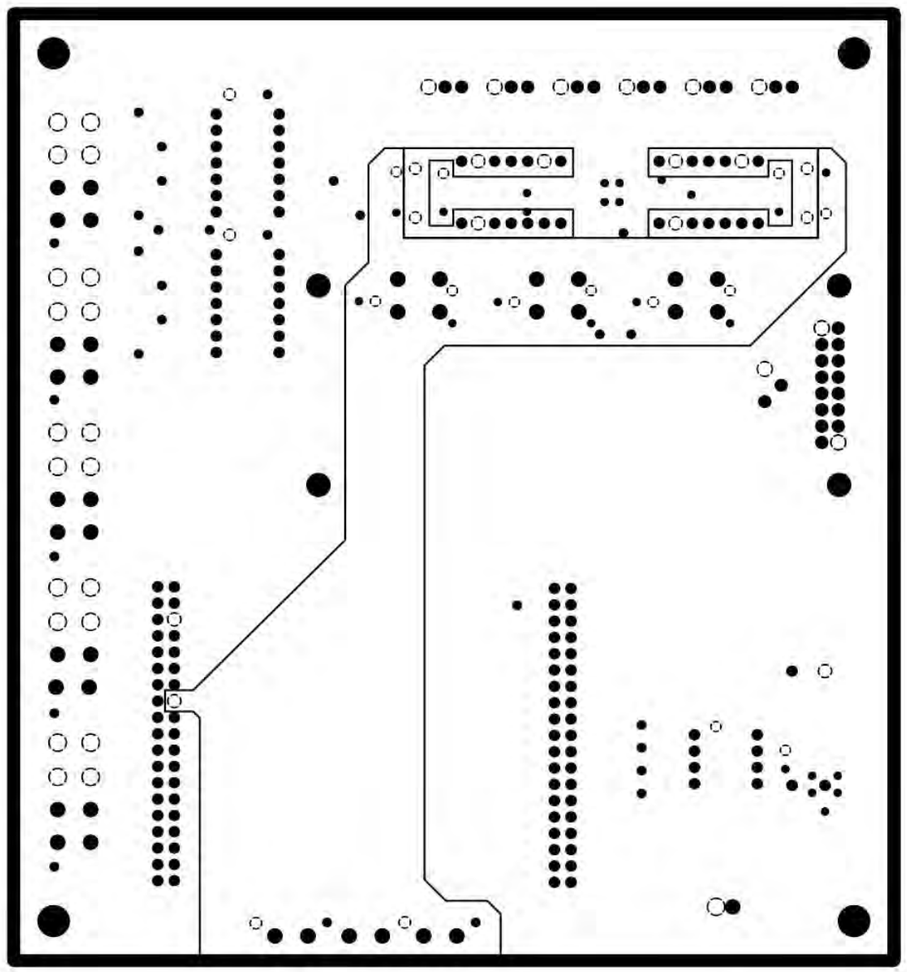
Universidad Complutense de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio "San Rafael (15) Transición a la Vida Adulta - Juego de Carreras.	TITULO Top Layer			
	ESCALA 1 / 1	FORMATO A3	HOJA 2/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



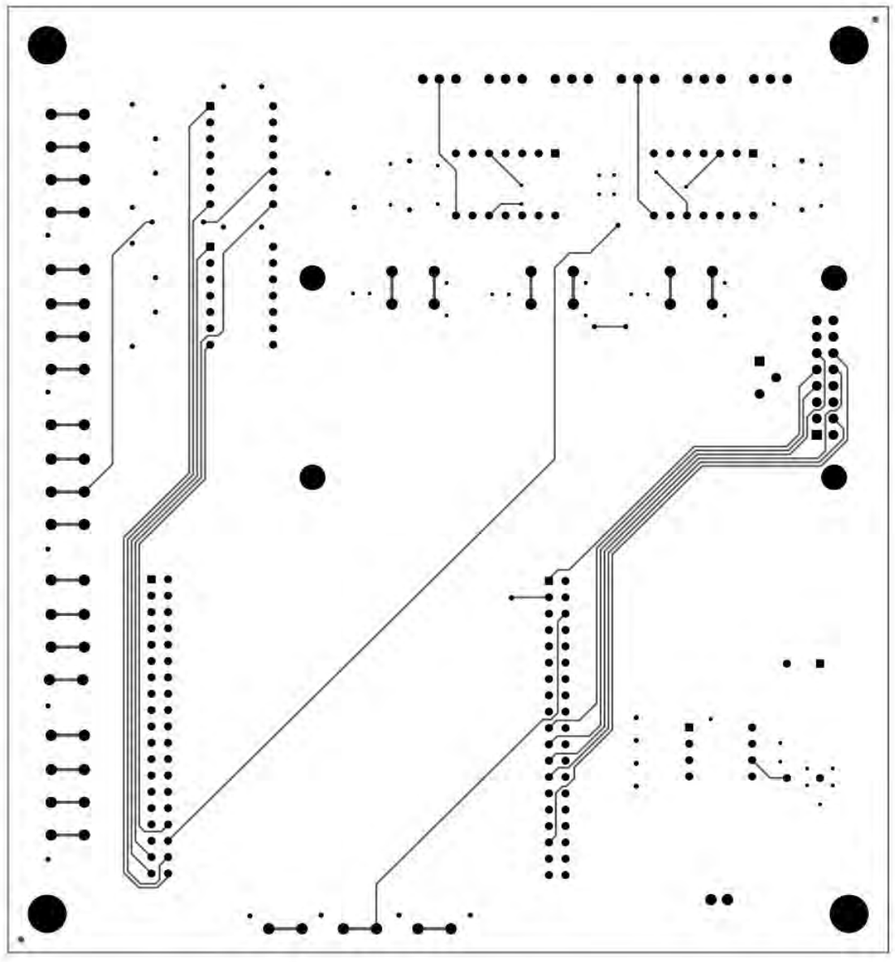
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15y Transición a la Vida Adulta - Juego de Carreras.	TITULO Internal Plane 1			
	ESCALA 1 / 1	FORMATO A3	HOJA 3/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



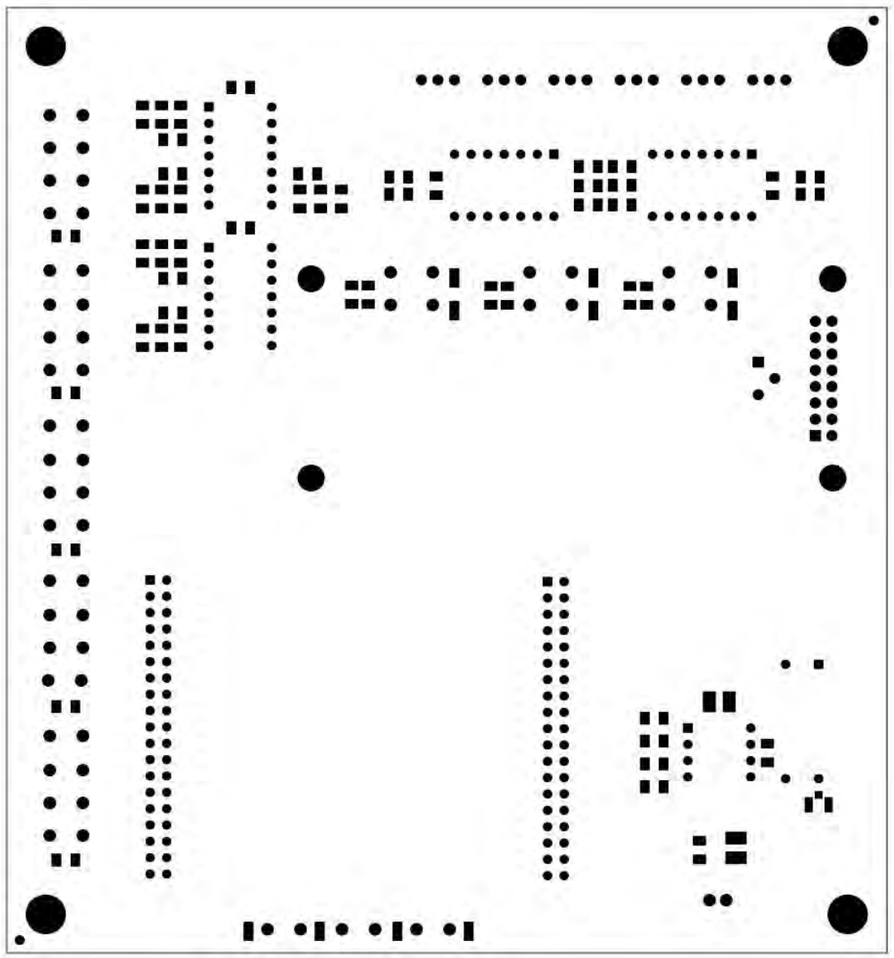
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15y) Transición a la Vida Adulta - Juego de Carreras.	TITULO Internal Plane 2			
	ESCALA 1 / 1	FORMATO A3	HOJA 4/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



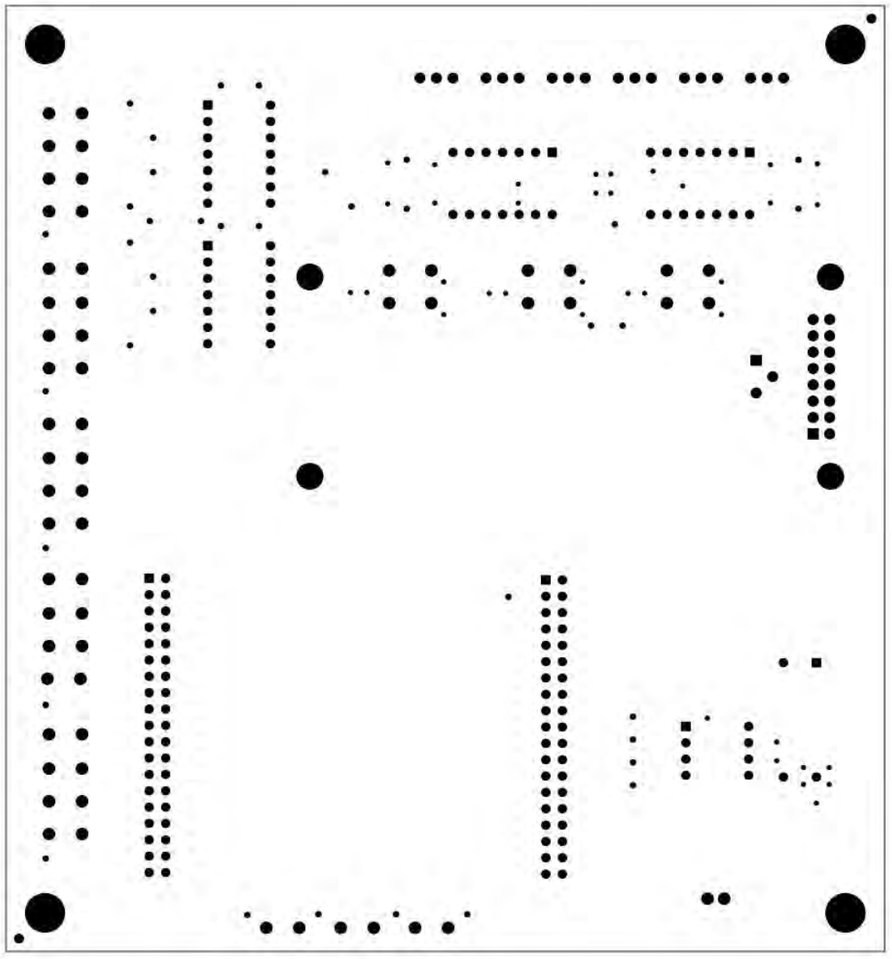
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juega de Carreras.	TITULO: Bottom Layer			
	ESCALA: 1 / 1	FORMATO: A3	HOJA: 5/11	

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15y. Transición a la Vida Adulta - Juego de Carreras.	TITULO Top Solder Mask Print		
	ESCALA 1 / 1	FORMATO A3	HOJA 6/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016

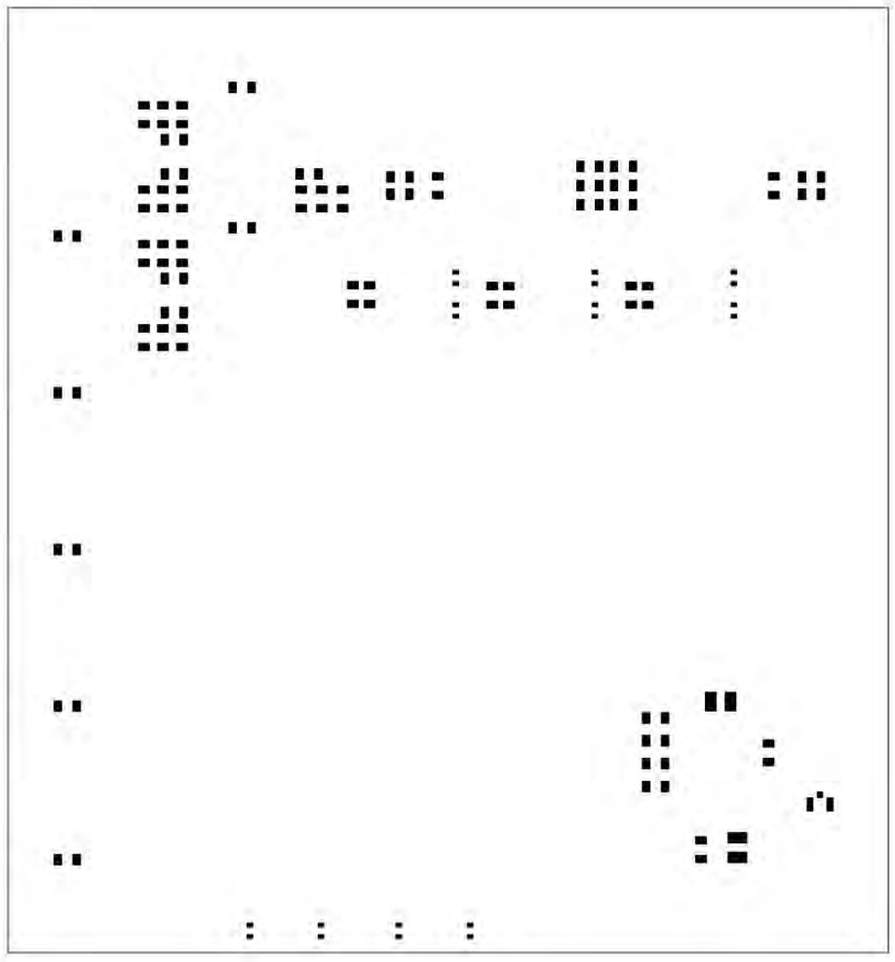


Universidad Carlos III de Madrid
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15).
Translación a la Vida Adulta - Juega de Carrera.

TITULO:
Bottom Solder Mask Print

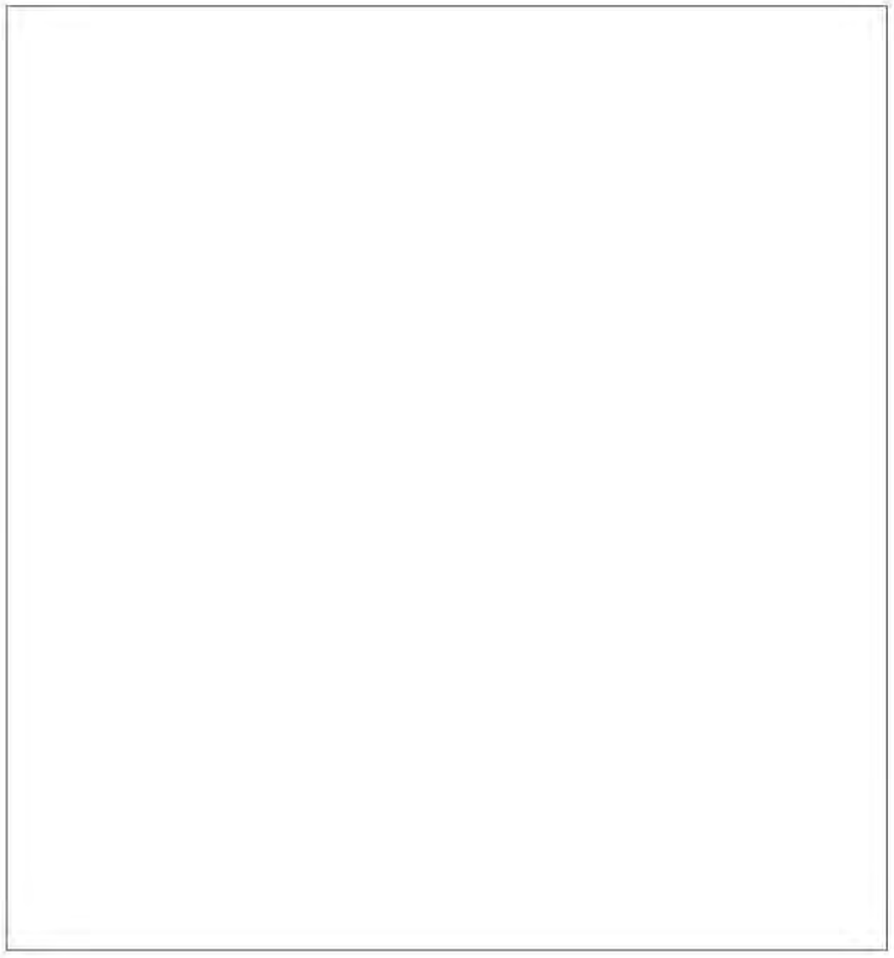
ESCALA:	FORMATO:	HOJA:
1 / 1	A3	7/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras.	TITULO Top Paste Mask Print		
	ESCALA 1 / 1	FORMATO A3	HOJA 8/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016

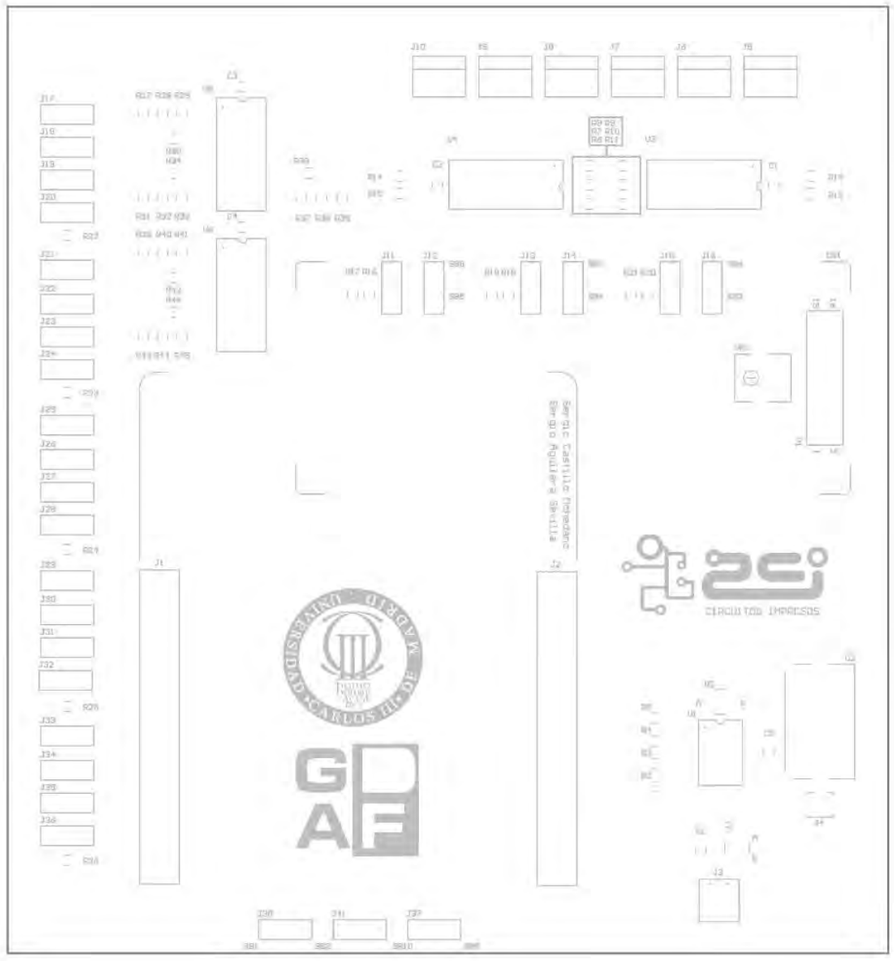


Universidad Carlos III de Madrid
Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras.

TITULO:
Bottom Paste Mask Print

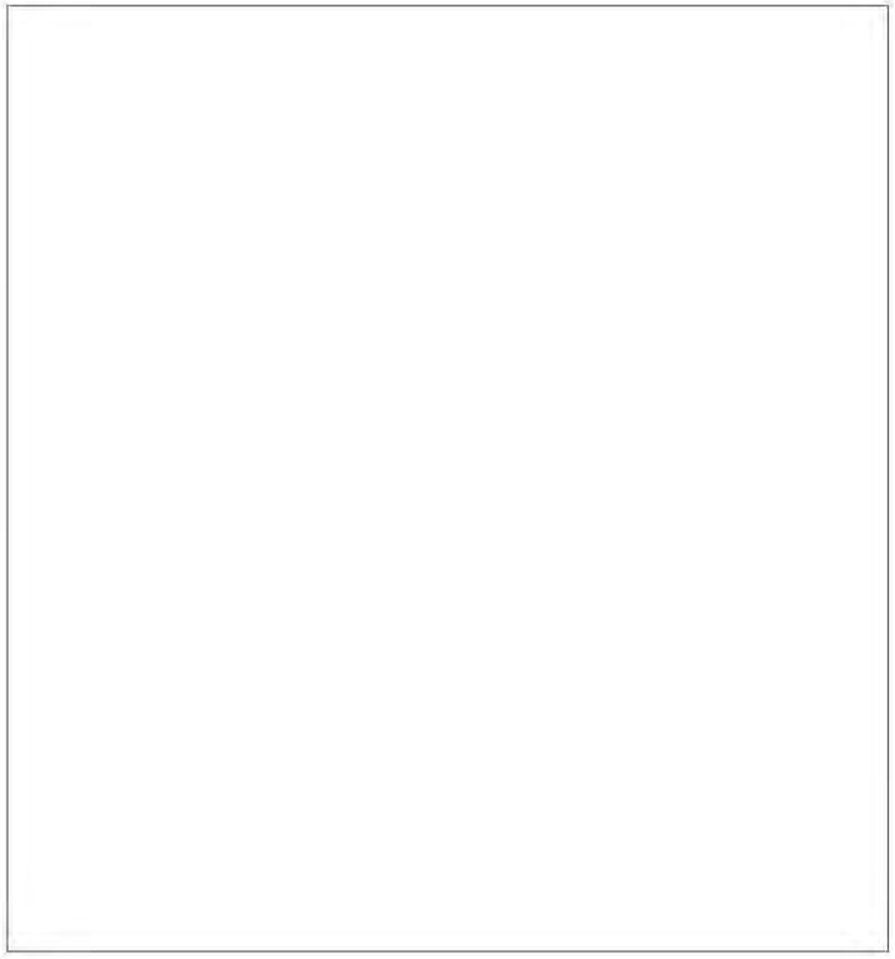
ESCALA:	FORMATO:	HOJA
1 / 1	A3	9/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta - Juego de Carreras.	TITULO Top Silkscreen Overlay		
	ESCALA 1 / 1	FORMATO A3	HOJA 10/11

DIBUJA	FECHA
Sergio Castillo	09/09/2016



Universidad Carlos III de Madrid

Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Traslación a la Vida Adulta - Juego de Carreras.

TITULO:

Bottom Silkscreen Overlay

ESCALA:	FORMATO:	HOJA
1 / 1	A3	11/11

- Firmware Bloque Carriles
  - main.c

```
 *   Created on: Aug 28, 2016
 *       Author: scastillom
 ******************************************************************************
 */


/* Includes ------------------------------------------------------------------*/
#include <stdlib.h>
#include <string.h>
#include "stm32f4xx_hal.h"
#include "steppermotors.h"
#include "fplayer.h"

#define MAXPOINTS 70
volatile uint16_t RxDataRF_Cnt;


/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart6;

uint8_t FPlayer_Buffer[LENGTH];
uint8_t RF_Buffer[LENGTH];
uint8_t character = '\0';
uint8_t nplayers = 0;
uint8_t players[4][16] =     {
                                        {"               \0"},
                                        {"               \0"},
                                        {"               \0"},
                                        {"               \0"}
                                    };
uint8_t i;
uint8_t j;
uint8_t characters[4][16] = {
                                        {"Personaje 1    \0"},
                                        {"Personaje 2    \0"},
                                        {"Personaje 3    \0"},
                                        {"Personaje 4    \0"}
                                    };
uint8_t player_Cnt = 0;
uint8_t pointPlayers[4] = {0,0,0,0};
int16_t max = -10;
uint8_t flag_Ended = 0;
uint8_t musicOn = 0;
uint8_t done = 0;


/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_USART6_UART_Init(void);

/* sendMotorsToStart
 */
```

```
057 void sendMotorsToStart();
058
059 /* deInitVariables
060  */
061 void deInitVariables();
062
063 int main(void)
064 {
065 /* MCU Configuration--------------------------------------------------------*/
066
067 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
068   HAL_Init();
069
070   /* Configure the system clock */
071   SystemClock_Config();
072
073   /* Initialize all configured peripherals */
074   MX_GPIO_Init();
075   MX_USART1_UART_Init();
076   MX_USART6_UART_Init();
077
078   while (1)
079   {
080         HAL_UART_Receive_IT(&huart6, RF_Buffer, LENGTH);
081
082         if (musicOn == 1)
083         {
084               buildCommand(PLAYMP3SONG_CMD, 17, FPlayer_Buffer);
085               HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
086               musicOn = 0;
087         }
088
089         if (musicOn == 2)
090         {
091               buildCommand(STOP_CMD, 0, FPlayer_Buffer);
092               HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
093               musicOn = 0;
094         }
095
096         if (RxDataRF_Cnt == LENGTH)
097         {
098               RxDataRF_Cnt = 0;
099               character = RF_Buffer[7];

100                //A = 1 jugador,
101                //B = 2 jugadores,
102                //C = 3 jugadores,
103                //D = 4 jugadores
104                if (character == 'A' || character == 'B' || character == 'C' ||
    character == 'D')
105                {
106                      switch (character)
107                      {
108                      case 'A':
109                            nplayers = 1;
110                            buildCommand(PLAYMP3SONG_CMD, 01, FPlayer_Buffer);
111                            HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
112                            break;
113                      case 'B':
114                            nplayers = 2;
```

```
115                                 buildCommand(PLAYMP3SONG_CMD, 02, FPlayer_Buffer);
116                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
117                                 break;
118                           case 'C':
119                                 nplayers = 3;
120                                 buildCommand(PLAYMP3SONG_CMD, 03, FPlayer_Buffer);
121                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
122                                 break;
123                           case 'D':
124                                 nplayers = 4;
125                                 buildCommand(PLAYMP3SONG_CMD, 04, FPlayer_Buffer);
126                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
127                                 break;
128                           }
129                     }
130
131                     //E = Personaje 1 (caballo),
132                     //F = Personaje 2 (dinosaurio),
133                     //G = Personaje 3 (coche),
134                     //H = Personaje 4 (moto)
135                     if (character == 'E' || character == 'F' || character == 'G' ||
    character == 'H')
136                     {
137                           switch (character)
138                           {
139                           case 'E':
140                                 strncpy(players[player_Cnt], characters[0], 16);
141                                 buildCommand(PLAYMP3SONG_CMD, 5, FPlayer_Buffer);
142                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
143                                 break;
144                           case 'F':
145                                 strncpy(players[player_Cnt], characters[1], 16);
146                                 buildCommand(PLAYMP3SONG_CMD, 6, FPlayer_Buffer);
147                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
148                                 break;
149                           case 'G':
150                                 strncpy(players[player_Cnt], characters[2], 16);
151                                 buildCommand(PLAYMP3SONG_CMD, 7, FPlayer_Buffer);
152                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
153                                 break;
154                           case 'H':
155                                 strncpy(players[player_Cnt], characters[3], 16);
156                                 buildCommand(PLAYMP3SONG_CMD, 8, FPlayer_Buffer);
157                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
158                                 break;
159                           }
160                           player_Cnt++;
161                           if (player_Cnt == nplayers)
162                           {
163                                 musicOn = 1;
164                                 player_Cnt = 0;
165                                 HAL_Delay(3000);
166                           }
```

```
167                 }
168
169                 //I = 10 puntos,
170                 //J = 15 puntos,
171                 //K = 20 puntos
172                 if(character == 'I' || character == 'J' || character == 'K')
173                 {
174                       player_Cnt++;
175
176                       if (players[player_Cnt - 1][10] == '1')
177                       {
178                             buildCommand(PLAYMP3SONG_CMD, 9, FPlayer_Buffer);
179                       }
180
181                       if (players[player_Cnt - 1][10] == '2')
182                       {
183                             buildCommand(PLAYMP3SONG_CMD, 10, FPlayer_Buffer);
184                       }
185
186                       if (players[player_Cnt - 1][10] == '3')
187                       {
188                             buildCommand(PLAYMP3SONG_CMD, 11, FPlayer_Buffer);
189                       }
190
191                       if (players[player_Cnt - 1][10] == '4')
192                       {
193                             buildCommand(PLAYMP3SONG_CMD, 12, FPlayer_Buffer);
194                       }
195
196                       switch (character)
197                       {
198                       case 'I':
199                             HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
200                             moveMotorDistance(player_Cnt, 10);
201                             pointPlayers[player_Cnt-1] = pointPlayers[player_Cnt-
    1] + 10;
202                             HAL_Delay(4000);
203                             buildCommand(PLAYMP3SONG_CMD, 18, FPlayer_Buffer);
204                             HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
205                             HAL_Delay(4000);
206                             break;
207
208                       case 'J':
209                             HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
210                             moveMotorDistance(player_Cnt, 15);
211                             pointPlayers[player_Cnt-1] = pointPlayers[player_Cnt-
    1] + 15;
212                             HAL_Delay(4000);
213                             buildCommand(PLAYMP3SONG_CMD, 19, FPlayer_Buffer);
214                             HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
215                             HAL_Delay(4000);
216                             break;
217
218                       case 'K':
219                             HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
220                             moveMotorDistance(player_Cnt, 20);
```

```
221                                 pointPlayers[player_Cnt-1] = pointPlayers[player_Cnt-
    1] + 20;
222                                 HAL_Delay(4000);
223                                 buildCommand(PLAYMP3SONG_CMD, 20, FPlayer_Buffer);
224                                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer,
    LENGTH);
225                                 HAL_Delay(4000);
226                                 break;
227                           }
228
229                           if (player_Cnt >= nplayers)
230                           {
231                                 player_Cnt = 0;
232                           }
233
234                           musicOn = 1;
235                     }
236
237                     //L = reset variables. Se reinicia el juego.
238                     if(character == 'L')
239                     {
240                           if (done == 0)
241                           {
242                                 sendMotorsToStart();
243                                 done = 1;
244                           }
245                           deInitVariables();
246                           musicOn = 2;
247                     }
248
249                     //M = Fin del juego, puntuación máxima alcanzada por uno de los
    jugadores.
250                     if(character == 'M')
251                     {
252                           flag_Ended = 1;
253                           for (i=0 ; i<nplayers ; i++)
254                           {
255                                 if (pointPlayers[i] >= max)
256                                 {
257                                       max = pointPlayers[i];
258                                 }
259                           }
260                     }
261               }
262
263               // chequea qué jugador consiguió mayor puntuación
264               // y manda a posición inicial los motores.
265               if (flag_Ended == 1)
266               {
267                     if (max == pointPlayers[3])
268                     {
269                           buildCommand(PLAYMP3SONG_CMD, 16, FPlayer_Buffer);
270                           HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
271                     }
272                     else if (max == pointPlayers[2])
273                     {
274                           buildCommand(PLAYMP3SONG_CMD, 15, FPlayer_Buffer);
275                           HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
276                     }
277                     else if (max == pointPlayers[1])
```

```
278                 {
279                       buildCommand(PLAYMP3SONG_CMD, 14, FPlayer_Buffer);
280                       HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
281                 }
282                 else if (max == pointPlayers[0])
283                 {
284                       buildCommand(PLAYMP3SONG_CMD, 13, FPlayer_Buffer);
285                       HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
286                 }
287
288                 HAL_Delay(3000);
289                 buildCommand(PLAYMP3SONG_CMD, 21, FPlayer_Buffer);
290                 HAL_UART_Transmit_IT(&huart1, FPlayer_Buffer, LENGTH);
291                 sendMotorsToStart();
292                 done = 1;
293                 flag_Ended = 0;
294           }
295     }
296
297   }
298
299   void deInitVariables()
300   {
301           for (i=0 ; i< LENGTH ; i++)
302           {
303                 FPlayer_Buffer[i] = ' ';
304                 RF_Buffer[i] = ' ';
305           }
306
307           character = '\0';
308           nplayers = 0;
309           for (i=0 ; i<4 ; i++)
310           {
311                 pointPlayers[i] = 0;
312                 for (j=0 ; j<16 ; j++)
313                 {
314                       players[i][j] = ' ';
315                       if (j == 15)
316                       {
317                             players[i][j] = '\0';
318                       }
319                 }
320           }
321           player_Cnt = 0;
322           max = -10;
323           flag_Ended = 0;
324           done = 0;
325
326           HAL_GPIO_WritePin(DIR_M1_GPIO_Port, DIR_M1_Pin, GPIO_PIN_RESET);
327           HAL_GPIO_WritePin(DIR_M2_GPIO_Port, DIR_M2_Pin, GPIO_PIN_RESET);
328           HAL_GPIO_WritePin(DIR_M3_GPIO_Port, DIR_M3_Pin, GPIO_PIN_RESET);
329           HAL_GPIO_WritePin(DIR_M4_GPIO_Port, DIR_M4_Pin, GPIO_PIN_RESET);
330   }
```

- fplayer.h

```
001  /*
002   * fplayer.h
003   *
004   *  Created on: Aug 28, 2016
005   *      Author: scastillom
006   */
007
008  #ifndef FPLAYER_H_
009  #define FPLAYER_H_
010
011
012  /* Includes ------------------------------------------------------------------*/
013  #include "stm32f4xx.h"
014
015  /* Defines -------------------------------------------------------------------*/
016  #define LENGTH                      10    //Data length to be sent.
017
018  #define STARTBYTE                   0x7E  //StartByte, '$S' by default.
019  #define VERSIONBYTE                 0xFF  //Version Information.
020  #define LENGTHBYTE                  0x06  //Number of bytes from CMD to Check_LSB,
021  #define FEEDBACKBYTE                0x00  //0x00 when transmitting.
022  #define ENDBYTE                     0xEF  //End Byte, '$O' by default.
023
024  /****************
025   *   Commands   *
026   ****************/
027
028  #define PLAYNEXT_CMD                0x01  //Play next track
029  #define PLAYPREV_CMD                0x02  //Play previous track
030  #define PLAYSONG_CMD                0x03  //Specify playback of a track 1-3000
031  #define TURNUPVOL_CMD               0x04  //Turn Up Volume
032  #define TURNDWNVOL_CMD              0x05  //Turn Down Volume
033  #define SETVOL_CMD                  0x06  //Specify volume Volume level:0-30
034  #define SETEQ_CMD                   0x07  //Specify EQ(0/1/2/3/4/5)
     0:Normal/1:Pop/2:Rock/3:Jazz/4:Classic/5:Bass
035  #define SINGLRPT_CMD                0x08  //Specify single repeat playback Tracks
     0001-3000
036  #define PLAYDEVICE_CMD              0x09  //Specify playback of a device(0/1)
     0:USB/1:SD // Specify playback source(0/1/2/3/4): U/TF/AUX/SLEEP/FLASH
037  #define SETSLEEP_CMD                0x0A  //Set Sleep
038  //#define NA_CMD                    0x0B  //N/A(Reserved Normal working)
039  #define RESET_CMD                   0x0C  //Reset
040  #define PLAY_CMD                    0x0D  //Play
041  #define PAUSE_CMD                   0x0E  //Pause
042  #define PLAYSNGFLDR_CMD             0x0F  //Specify playback a track in a folder
     Folders 01-99
043  #define SETAMP_CMD                  0x10  //Audio amplification setting
     MSB=1:amplifying on, LSB:set gain 0-31
044  #define REPEATALL_CMD               0x11  //Set all repeat playback 1:start all
     repeat playback; 0:stop playback
045  #define PLAYMP3SONG_CMD             0x12  //Specify playback of folder named "MP3"
046  #define ADVERTCUT_CMD               0x13  //Inter cut an advertisement
047  #define PLYTRCKFLDR_CMD             0x14  //Specify playback a track in a folder
     that supports 3000 tracks Supports 15 folders only(01-15)
048  #define STOPADVERT_CMD              0x15  //Stop playing inter-cut advertisement
     and go back to play the music interrupted
049  #define STOP_CMD                    0x16  //Stop
050  #define REPEATFLDR_CMD              0x17  //Specify repeat playback of a folder
```


```
051 #define RNDMPLAY                    0x18  //Set random playback
052 #define REPEAT_CMD                  0x19  //Set repeat playback of current track
053 #define SETDAC_CMD                  0x1A  //Set DAC
054
055 /*********************
056  *   Query Commands  *
057  *********************/
058
059 //#define NA_CMD                    0x3C  //N/A(Reserved)
060 //#define NA_CMD                    0x3D  //N/A(Reserved)
061 //#define NA_CMD                    0x3E  //N/A(Reserved)
062 #define QRYDEVIDE_CMD               0x3F  //Query current online storage device
063 #define QRYERROR_CMD                0x40  //Module returns an error data with this
    command
064 #define QRYFDBCKDBG_CMD             0x41  //Module reports a feedback with this
    command // "debug mode"
065 #define QRYSTATUS_CMD               0x42  //Query current status
066 #define QRYVOL_CMD                  0x43  //Query current volume
067 #define QRYEQ_CMD                   0x44  //Query current EQ
068 #define QRYPLAYBACK_CMD             0x45  //Query the current playback mode
069 #define QRYVERSION_CMD              0x46  //Query the current software version
070 #define QRYQTYUSB_CMD               0x47  //Query total file numbers of USB flash
    disk
071 #define QRYQTYSD_CMD                0x48  //Query total file numbers of micro SD
    Card
072 #define QRYQTYFLSH_CMD              0x49  //Query the total number of flash files
073 //#define NA_CMD                    0x4A  //N/A(Reserved)
074 #define QRYTRCKUSB_CMD              0x4B  //Query current track of USB flash disk
075 #define QRYTRCKSD_CMD               0x4C  //Query current track of micro SD Card
076 #define QRYTRCKFLSH_CMD             0x4D  //Queries the current track of Flash
077 #define QRYFILESINFOLDER_CMD        0x4E  //Query total file numbers of a folder
078 #define QRYFOLDERLNMBR_CMD          0x4F  //Query total folder numbers of the
    storage device
079
080
081 /* Function Prototypes ----------------------------------------------------*/
082
083 /* putChr
084  */
085 void putChr (uint8_t Word, uint8_t Indx, uint8_t Buff[]);
086
087 /* buildCommand
088  */
089 void buildCommand (uint8_t Word, uint16_t Param, uint8_t Buff[]);
```

- fplayer.c

```
001  /*
002   * fplayer.c
003   *
004   *  Created on: Aug 28, 2016
005   *      Author: scastillom
006   */
007
008  /* Includes ------------------------------------------------------------------*/
009  #include "stm32f4xx.h"
010  #include "fplayer.h"
011
012  /** @putChr
013    * @brief  Pone un 1 byte en la posición Index del buffer
014    * @param  Word: Byte de información que se desea indexar en el buffer
015    * @param  Indx: índice del buffer, cada byte se guarda en un espacio
016    *         del buffer distinto.
017    * @param  Buff[]: Buffer.
018    * @retval none
019    */
020  void putChr(uint8_t Word, uint8_t Indx, uint8_t Buff[])
021  {
022        Buff[Indx] = Word;
023  }//putChr
024
025  /** @buildCommand
026    * @brief  Construye el dato que se va a enviar al FPlayer para que
027    *         este lo entienda correctamente
028    * @param  Word: Byte de información que se desea indexar en el buffer.
029    * @param  Buff[]: Buffer.
030    * @retval none
031    */
032  void buildCommand(uint8_t Word, uint16_t Param, uint8_t Buff[])
033  {
034        uint16_t xorsum = 0;
035        uint8_t i;
036
037        putChr(STARTBYTE, 0, Buff);
038        putChr(VERSIONBYTE, 1, Buff);
039        putChr(LENGTHBYTE, 2, Buff);
040        putChr(Word, 3, Buff);
041        putChr(FEEDBACKBYTE, 4, Buff);
042        putChr((uint8_t)(Param >> 8), 5, Buff);
043        putChr((uint8_t)(Param & 0x00ff), 6, Buff);
044
045        for(i=1;i<7;i++)  //CHECKSUM = 0 -(VERSIONBYTE + LENGTHBYTE + COMMAND
046        {                 // + FDBCK + PARAM_MSB + PARAM_LSB) =0xHHhh
047              xorsum = xorsum + Buff[i];
048        }
049        xorsum = 0 - xorsum;
050
051        putChr((uint8_t)(xorsum >> 8), 7, Buff);
052        putChr((uint8_t)(xorsum & 0x00ff), 8, Buff);
053        putChr(ENDBYTE, 9, Buff);
054  }//buildCommand
```


- steppermotors.h

```
/*
 * steppermotors.h
 *
 *  Created on: Aug 23, 2016
 *      Author: scast
 */

#ifndef STEPPERMOTORS_H_
#define STEPPERMOTORS_H_

/* Includes ------------------------------------------------------------------*/
#include <math.h>
#include "stm32f4xx_hal.h"
#include <stdio.h>



/* Function Prototypes -----------------------------------------------------*/

/* moveMotorDegrees
 */
void moveMotorDegrees (uint8_t player, uint16_t degrees);

/* changeDir
 */
void changeDir (uint8_t player);

/* moveMotorDistance
 */
void moveMotorDistance (uint8_t player, double centimeters);


#endif /* STEPPERMOTORS_H_ */
```

- steppermotors.c

```
/*
 * steppermotors.c
 *
 *  Created on: Aug 23, 2016
 *      Author: scastillom
 */

#include "steppermotors.h"

 volatile uint16_t MA0_Flag = 0;
 volatile uint16_t MA1_Flag = 0;
 volatile uint16_t MA2_Flag = 0;
 volatile uint16_t MAAux_Flag = 0;

 /** @putChr
   * @brief  Activa el movimiento del motor del jugador correspondiente
   *         un número de pasos equivalente a los grados que se desee.
   * @param  player: jugador actual, en función de este valor se mueve
   *         uno de los cuatro motores.
   * @param degrees: grados que se desea mover el motor.
```

```
021       * @retval none
022       */
023   void moveMotorDegrees (uint8_t player, uint16_t degrees)
024   {
025         double steps = 0;
026         uint16_t cycles = 0;
027         uint16_t i = 0;
028         uint8_t stop = 0;
029         double oneLapSteps = 200;
030         double oneLapDegrees = 360;
031
032
033         steps = degrees * (oneLapSteps/oneLapDegrees);
034         cycles = round(2 * steps);
035
036         switch (player)
037         {
038         case 1:
039               HAL_GPIO_WritePin(RESET_M1_GPIO_Port, RESET_M1_Pin, GPIO_PIN_SET);
040               break;
041         case 2:
042               HAL_GPIO_WritePin(RESET_M2_GPIO_Port, RESET_M2_Pin, GPIO_PIN_SET);
043               break;
044         case 3:
045               HAL_GPIO_WritePin(RESET_M3_GPIO_Port, RESET_M3_Pin, GPIO_PIN_SET);
046               break;
047         case 4:
048               HAL_GPIO_WritePin(RESET_M4_GPIO_Port, RESET_M4_Pin, GPIO_PIN_SET);
049               break;
050         }
051
052
053         for (i = 0; i < cycles; i++)
054                {
055                      HAL_Delay(5);
056                      switch (player)
057                          {
058                          case 1:
059                                if (stop == 0)
060                                {
061                                      HAL_GPIO_TogglePin(STEP_M1_GPIO_Port,
      STEP_M1_Pin);
062                                }
063                                if ((MA2_Flag == 1 && MA1_Flag == 1 && MA0_Flag ==
      1) || (MA2_Flag == 1 && MA1_Flag == 1 && MA0_Flag == 0))
064                                {
065                                      stop = 1;
066                                      HAL_GPIO_WritePin(STEP_M1_GPIO_Port,
      STEP_M1_Pin, GPIO_PIN_RESET);
067                                }
068                                break;
069                          case 2:
070                                if (stop == 0)
071                                {
072                                      HAL_GPIO_TogglePin(STEP_M2_GPIO_Port,
      STEP_M2_Pin);
073                                }
074                                if ((MA2_Flag == 1 && MA1_Flag == 0 && MA0_Flag ==
      0) || (MA2_Flag == 1 && MA1_Flag == 0 && MA0_Flag == 1))
075                                {
```


```
076                                         stop = 1;
077                                         HAL_GPIO_WritePin(STEP_M2_GPIO_Port,
    STEP_M2_Pin, GPIO_PIN_RESET);
078                                   }
079                                   break;
080                             case 3:
081                                   if (stop == 0)
082                                   {
083                                         HAL_GPIO_TogglePin(STEP_M3_GPIO_Port,
    STEP_M3_Pin);
084                                   }
085                                   if ((MA2_Flag == 0 && MA1_Flag == 1 && MA0_Flag ==
    0) || (MA2_Flag == 0 && MA1_Flag == 1 && MA0_Flag == 1))
086                                   {
087                                         stop = 1;
088                                         HAL_GPIO_WritePin(STEP_M3_GPIO_Port,
    STEP_M3_Pin, GPIO_PIN_RESET);
089                                   }
090                                   break;
091                             case 4:
092                                   if (stop == 0)
093                                   {
094                                         HAL_GPIO_TogglePin(STEP_M4_GPIO_Port,
    STEP_M4_Pin);
095                                   }
096                                   if ((MA2_Flag == 0 && MA1_Flag == 0 && MA0_Flag ==
    1) || (MAAux_Flag == 1))
097                                   {
098                                         stop = 1;
099                                         HAL_GPIO_WritePin(STEP_M4_GPIO_Port,
    STEP_M4_Pin, GPIO_PIN_RESET);
100                                   }
101                                   break;
102                             }
103                 }
104
105         stop = 0;
106         MAAux_Flag = 0;
107         MA2_Flag = 0;
108         MA1_Flag = 0;
109         MA0_Flag = 0;
110
111         HAL_Delay(200);
112         switch (player)
113         {
114         case 1:
115               HAL_GPIO_WritePin(RESET_M1_GPIO_Port, RESET_M1_Pin, GPIO_PIN_RESET);
116               break;
117         case 2:
118               HAL_GPIO_WritePin(RESET_M2_GPIO_Port, RESET_M2_Pin, GPIO_PIN_RESET);
119               break;
120         case 3:
121               HAL_GPIO_WritePin(RESET_M3_GPIO_Port, RESET_M3_Pin, GPIO_PIN_RESET);
122               break;
123         case 4:
124               HAL_GPIO_WritePin(RESET_M4_GPIO_Port, RESET_M4_Pin, GPIO_PIN_RESET);
125               break;
126         }
127   }//moveMotorDegrees
128
129
```

```
130   /** @changeDir
131     * @brief  Cambia la dirección del motor correspondiente al jugador
132     *         un número de pasos equivalente a los grados que se desee.
133     * @param  player: jugador actual, en función de este valor se actúa
134     *         sobre uno de los motores..
135     * @retval none
136     */
137   void changeDir (uint8_t player)
138   {
139         switch(player)
140         {
141         case 1:
142               HAL_GPIO_TogglePin(DIR_M1_GPIO_Port, DIR_M1_Pin);
143               break;
144         case 2:
145               HAL_GPIO_TogglePin(DIR_M2_GPIO_Port, DIR_M2_Pin);
146               break;
147         case 3:
148               HAL_GPIO_TogglePin(DIR_M3_GPIO_Port, DIR_M3_Pin);
149               break;
150         case 4:
151               HAL_GPIO_TogglePin(DIR_M4_GPIO_Port, DIR_M4_Pin);
152               break;
153         }
154   }//changeDir
155
156   /** @putChr
157     * @brief  Activa el movimiento del motor del jugador correspondiente
158     *         un número de pasos equivalente a los centímetros que se desee.
159     * @param  player: jugador actual, en función de este valor se mueve
160     *         uno de los cuatro motores.
161     * @param  centimeters: centímetros que se desea mover el motor.
162     * @retval none
163     */
164   void moveMotorDistance (uint8_t player, double centimeters)
165   {
166         double pi = 3.14159265;
167         double radius = 2.5;          //en centimetros
168         double perimeter;
169         uint16_t degrees;
170         perimeter = 2*pi*radius;      //en centimetros
171
172         //360 degrees is equal to perimeter
173         degrees = round((centimeters * 360) / perimeter);
174         moveMotorDegrees (player, degrees);
175   }//moveMotorDegrees
```

- stm32f4xx_it.c

```
  *   Created on: Aug 28, 2016
  *       Author: scastillom
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
#include "stm32f4xx.h"
#include "stm32f4xx_it.h"


/* External variables --------------------------------------------------------*/
extern UART_HandleTypeDef huart1;
extern UART_HandleTypeDef huart6;

/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers         */
/******************************************************************************/

/**
* @brief This function handles System tick timer.
*/
void SysTick_Handler(void)
{
  HAL_IncTick();
  HAL_SYSTICK_IRQHandler();
}

/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************/

/**
* @brief This function handles EXTI line0 interrupt.
*/
void EXTI0_IRQHandler(void)
{
        if(__HAL_GPIO_EXTI_GET_IT(MAAux_Pin) != RESET)
        {
              __HAL_GPIO_EXTI_CLEAR_IT(MAAux_Pin);
              MAAux_Flag = 1;
        }
}

/**
* @brief This function handles EXTI line1 interrupt.
*/
void EXTI1_IRQHandler(void)
{
        if(__HAL_GPIO_EXTI_GET_IT(MA0_Pin) != RESET)
        {
              __HAL_GPIO_EXTI_CLEAR_IT(MA0_Pin);
              MA0_Flag = 1;
        }
}
```

```
059
060 /**
061 * @brief This function handles EXTI line2 interrupt.
062 */
063 void EXTI2_IRQHandler(void)
064 {
065         if(__HAL_GPIO_EXTI_GET_IT(MA1_Pin) != RESET)
066         {
067               __HAL_GPIO_EXTI_CLEAR_IT(MA1_Pin);
068               MA1_Flag = 1;
069         }
070 }
071
072 /**
073 * @brief This function handles EXTI line3 interrupt.
074 */
075 void EXTI3_IRQHandler(void)
076 {
077         if(__HAL_GPIO_EXTI_GET_IT(MA2_Pin) != RESET)
078         {
079               __HAL_GPIO_EXTI_CLEAR_IT(MA2_Pin);
080               MA2_Flag = 1;
081         }
082 }
083
084 /**
085 * @brief This function handles USART1 global interrupt.
086 */
087 void USART1_IRQHandler(void)
088 {
089   HAL_UART_IRQHandler(&huart1);
090 }
091
092 /**
093 * @brief This function handles USART6 global interrupt.
094 */
095 void USART6_IRQHandler(void)
096 {
097   HAL_UART_IRQHandler(&huart6);
098   RxDataRF_Cnt++;
099 }
```

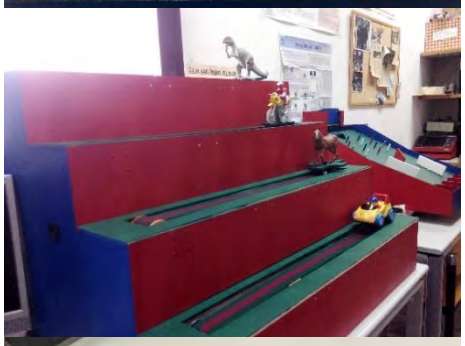

- Manual de Usuario

JUEGO DE CARRERAS ADAPTADO

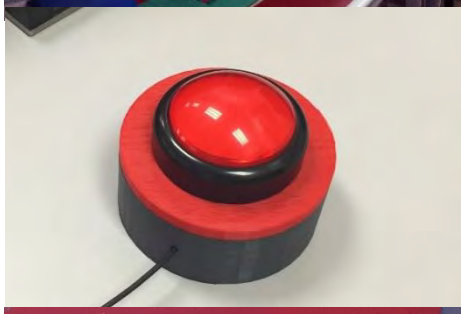
DESCRIPCIÓN DE COMPONENTES

El juego de carreras está compuesto por los siguientes elementos:

El Bloque Rampa: Desde el cual se efectúa todo el control (selección de jugadores, selección de personajes…) a través del panel de control situado en el lateral. Una vez que se ha terminado de configurar la partida, basta con pulsar un botón para accionar la apertura de la trampilla, que hará que la bola caiga en una de las cinco casillas de la parte inferior de la rampa y que hará que se mueva el personaje del jugador que en ese momento esté tirando.



El **Bloque Carriles**: Compuesto por cuatro carriles donde pueden jugar hasta cuatro jugadores. Durante toda la partida se produce un barrido auditivo de todo lo que está sucediendo gracias a los dos altavoces que hay situados en cada lateral.



Botón Rojo: Se conecta a un terminal JACK hembra de 3.5mm situado en el lateral de la rampa. Es el encargado del accionamiento de la trampilla. Al terminal JACK se le puede conectar otro pulsador más conveniente para el jugador, si fuera necesario. Y se pueden cambiar los pulsadores en medio de la partida para que cada jugador use el pulsador que mejor le convenga.

Muñecos: Hay cuatro muñecos distintos en función del personaje que se elija: un dinosaurio, un caballo, un coche y una moto. Cada uno de los muñecos tiene una base con un imán en la parte inferior de la misma. Las cintas transportadoras tienen otra base con otro imán. Es posible intercambiar los muñecos entre los distintos jugadores en función del personaje que haya elegido cada uno.

MONTAJE

Cada Bloque es independiente y pueden estar separados, aunque ambos deben estar conectados para poder jugar. Ambos tienen un enchufe y un botón como el de la siguiente figura. Para detectar que el bloque está enchufado y conectado, el botón de alimentación debe estar encendido.

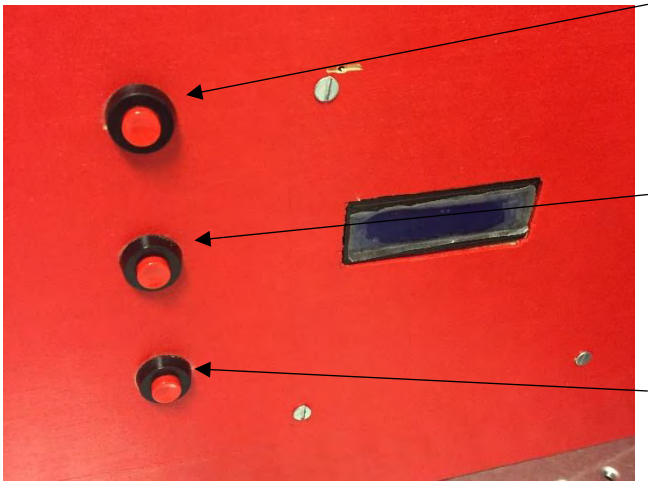
Ambos bloques deben estar en la misma habitación y los dos deben ser visibles por parte de los jugadores para disfrutar plenamente de la experiencia.

Aunque el orden en que se enciendan los dos bloques no influye en el funcionamiento del juego, se recomienda encender primero el Bloque Carriles y a continuación el Bloque Rampa. Si se hace al contrario se recomienda apretar antes de empezar a jugar el botón RESET del panel de control del Bloque Rampa.

Es requisito indispensable antes de empezar la configuración desde el panel de control que ambos bloques estén encendidos.

CONFIGURACIÓN

El panel de control dispone de 3 botones y un display.



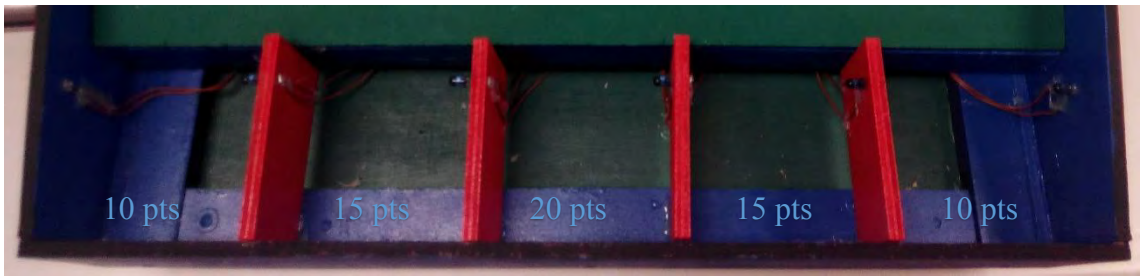
BOTÓN **MOVER**: Es el botón situado en la parte superior de la pantalla. Mediante este botón nos desplazamos a través de las distintas opciones que nos ofrece el menú de la pantalla.
BOTÓN **OK**: Botón central mediante el cual se selecciona la opción que se desea elegir. También es el botón que hay que presionar antes de habilitar una nueva pulsación del botón de accionamiento.
BOTÓN **RESET:** Botón habilitado para poder reiniciar la partida en cualquier momento si fuera necesario.

Para configurar una nueva partida se deben seguir los siguientes pasos:

1. El primer mensaje que aparecerá en la pantalla será "*Seleccione número de jugadores:"*. Inmediatamente pondrá "*número de jugadores: 4"*. Para cambiar el número de jugadores presione el BOTÓN MOVER. Cuando esté de acuerdo con su elección presione el BOTÓN OK.
2. A continuación, aparecerá un mensaje en la pantalla: "*Seleccione Personajes"*. Seleccione qué personaje desea que tenga cada jugador en la partida. Y en función de su elección coloque los muñecos en su posición correspondiente.
3. Comienza la partida. En la pantalla aparece un mensaje: "*Jugador 1: Presione Botón Rojo"*. En este momento comienza además a sonar la música característica que indica que la configuración de la partida ya ha terminado y la partida ha comenzado.

JUEGO

El objetivo del juego es que el jugador llegue al final de la carrera antes que los demás jugadores. Para ello la bola debe caer en una de las cinco casillas de la parte inferior de la rampa. Cada una de las casillas tiene una puntuación en función de la cual se moverá el personaje una distancia proporcional.



La puntuación necesaria para ganar la partida es de 70 puntos. La longitud de la cinta transportadora sobre la que se desplazan los carriles es de 70 cm. Es decir, cada punto conseguido corresponde a un centímetro. Si el jugador consigue 20 puntos, el personaje suyo se desplazará 20 centímetros.

Con tal de que todos los jugadores tengan las mismas oportunidades de ganar, se establecen las siguientes condiciones:

- Si en una misma ronda, dos o más jugadores llegan al final de la carrera. Ganará aquel que hubiera conseguido una mayor puntuación.
- Si dos o más jugadores llegan al final de la carrera con la misma puntuación, ganará el último de ellos que hubiera logrado esa puntuación.
- El ganador se establece al final de cada ronda. Si hay cuatro jugadores, los cuatro jugadores tienen la oportunidad de jugar en esa ronda, aunque el primero hubiera llegado ya a meta.

Cuando empieza el juego:

- Es el turno del primer jugador, que deberá haber colocado su pulsador conveniente antes de terminar la configuración de la partida.
- Cuando el primer jugador haya terminado su turno y se haya movido su personaje, se debe volver a colocar la pelota en la caseta de la rampa.
- Antes de que el siguiente jugador empiece su turno, se debe pulsar el BOTÓN OK en el panel de control para habilitar la pulsación del siguiente jugador. Si el siguiente jugador tiene que cambiar su pulsador, es imprescindible que lo haga antes de pulsar el BOTÓN OK, ya que de lo contrario el accionamiento de la trampilla se puede producir de forma inesperada.
- Este proceso se repite continuamente hasta que uno de los jugadores gane la partida. Cuando termina el juego, los personajes vuelven automáticamente a su posición inicial. En este momento se muestra en la pantalla del panel de control el jugador que ha ganado y su puntuación. Para volver a empezar otra partida se debe pulsar el BOTÓN RESET.
- Si se desea interrumpir el transcurso de juego antes de que finalice, basta con pulsar el BOTÓN RESET. Los personajes vuelven automáticamente a su posición inicial. Si no lo hicieran, se pueden desplazar fácilmente con la mano.

GUÍA RÁPIDA ANTE POSIBLES INCIDENCIAS.

- **Cuando cae la pelota, la casilla no ha detectado su paso y en la pantalla continúa saliendo el mensaje de "*Bola cayendo, buena suerte.*"**

Es posible que en este caso alguno de los sensores de las casillas no esté correctamente alineado. Asegura que todas las cabezas de los leds estén en ángulo recto con la madera.

- **Durante la configuración de la partida, la pantalla se ha quedado congelada y ni el BOTÓN OK ni el BOTÓN MOVER responden.**

En este caso, la solución está en pulsar el BOTÓN RESET y volver a configurar la partida.

- **Durante el intercambio del pulsador para el turno del siguiente jugador, se ha accionado la apertura de la trampilla antes incluso de conectar el siguiente pulsador.**

Suele ocurrir cuando el BOTÓN OK se ha pulsado antes de cambiar el pulsador. Hay que asegurarse de conectar el nuevo pulsador antes de presionar el BOTÓN OK.

- **La partida aparentemente ya ha llegado a su fin, pero no ocurre nada.**

Del mismo modo que cada vez que se cambia de turno hay que pulsar el BOTÓN OK, también hay que hacerlo al final de esta última ronda.

- **El interruptor ON/OFF del Bloque Carriles no se enciende. No sé si el Bloque Carriles está conectado o no.**

Si el botón de ON/OFF no se enciende, una forma de comprobar si el Bloque Carriles está realmente encendido es mirar a través de las cintas transportadoras hacia el interior del bloque. Si está encendido se observarán una serie de luces en la placa de circuito impreso.